A Formal Language Theoretic Approach to Self–Organizing Networks

A Dissertation Submitted in Partial Fulfillment
of the Requirements for the Degree of
Doctor of Philosophy
in
Computer Science

by

Katalin Anna Lázár

Eötvös Loránd University
PhD School in Computer Science
Director: **Dr. János Demetrovics**
PhD Program in Information Systems
Director: **Dr. András Benczúr**

Supervisors:

Dr. Erzsébet Csuhaj–Varjú
Computer and Automation Research Institute of the Hungarian Academy of Sciences and
Department of Algorithms and Applications, Eötvös Loránd University

Dr. András Lőrincz
Department of Software Technology and Methodology, Eötvös Loránd University

Budapest, 2010

Abstract

The concept and the reality of self–organizing networks have come to pervade modern society. But what exactly is a self–organizing network? Scientists from a range of disciplines have been pursuing questions on the particularities of self–organizing networks. Our work addresses self–organizing systems that compile to the scale–free small world model. We model self–organizing networks at syntactical level as well as reveal some semantical and experimental aspects related to them. At syntactical level, we use devices from grammar systems theory: in grammar systems theory the agents are represented by grammars and the generated strings describe the behaviour of the system. At experimental level, we utilize the methods of selective learning and value estimation under evolutionary pressure. The selection is influenced by the ever changing external world and by the competing individuals. First, we model peer–to–peer networks with the aid of networks of parallel multiset string processors. We establish the connection between the growth of the number of strings being present during the computation at the components of these networks of parallel multiset string processors and the growth function of a developmental system. We formalize security rules that conform to self–organizing dynamic systems and allow intra– and intercommunity collaborations. Our approach guarantees quick and efficient local analysis of the security requirements, thus reducing the need for global verification. Secondly, we illustrate the great diversity of employing regulated rewriting devices in eco–grammar systems to describe the search strategy of Internet crawlers. We prove that if we ignore the aging of the web pages in the model, then systems with rather simple component grammars suffice to identify any recursively enumerable language. Whereas if the web pages may become obsolete, then the efficiency of the cooperation of the agents decreases considerably. We also examine the extent to which communication makes a goal–oriented community efficient in different graph topologies through simulations. Finally, we extend the conditions of dynamic team constitution in eco–grammar systems to capture the behaviour of agents participating in network cluster formation. From the language classes that these systems are capable of generating, we deduce the difficulty of the problem they can solve. Depending on the team constitution mode, different classes of languages can be obtained. For all self–organizing networks to be presented in this dissertation, we also propose some further research directions.

Acknowledgements

I am grateful to Dr. Erzsébet Csuhaj–Varjú and Dr. András Lőrincz for advising me during my PhD studies and for introducing grammar systems theory and various fields of artificial intelligence to me. I am also indebted to them for their valuable comments and suggestions on this work.

I would like to express my thanks to Dr. Csilla Farkas for the chance to work with her on information security, for her guidance and assistance with my research.

I owe thanks to John von Neumann Computer Society for aiding my studies with their scholarship.

Last but not least, thanks to my family for supporting me to finish this work.

Table of Contents

List of Figures

Chapter 1

Introduction

In this dissertation we provide a formal language theoretic approach to self–organizing networks. But what exactly is a self–organizing network? How can formal language theoretic tools can be applied to capture the characteristics of self–organizing networks? To this end, in the current chapter we set the stage by presenting a selection of works that we feel are important historical antecedents to contemporary research.

1.1 Self–Organizing Networks

The theory of self–organizing systems is an interdisciplinary area, embracing works from the field of cybernetics and systems theory [7, 8, 47], physics [10, 55, 56, 97], biology [67, 68], mathematics [116, 130], computer science [60, 70] and economics [24]. Despite the ubiquitousness of self–organization, never has a unified view been formulated on the traits of the systems that can be characterized by this phenomenon.

The theory of self–organizing systems originated in the circles of cyberneticians and system theorists of the period after the Second World War. In effect, the term *self–organizing system* first appeared in print in 1947 in [7]. Ashby claims that every isolated determinate dynamical system, regardless of the nature of its initial state, tends to evolve towards an

equilibrium state [8], or in other words to an attractor [10]. The evolution towards the equilibrium state can be interpreted as selection.

Heinz von Foerster introduced the principle of *order from noise* [47]. He asserts that the high degree of random perturbations or *noise* accelerates self–organization, the production of *order* in the system.

Stuart Kauffman studies the abundance of self–organizing structures in nature [67, 68]. He concludes that complexity itself leads to self–organization. According to Kauffman, self–organizing systems tend to reside on the *edge of chaos*, i.e. the narrow domain between stability and turbulent, chaotic activity [67, 68]. The term *self–organized criticality* was coined by Per Bak to describe the mechanism that keeps self–organizing systems on this critical edge [10].

Francis Heylighen characterizes self–organizing systems by distributed control, by global order resulting from local interactions, by robustness, by resilience and by transition from a positive feedback phase to a negative one (and vice versa) through an equilibrium state [55, 56]. In self–organizing systems there is a range of stable states or configurations into which the system may settle. The number of stable states ought not to be excessively high lest the evolution become uncontrollably chaotic. The most adequate configurations are selected on the basis of adaptivity, either directly by the environment or indirectly by the subsystems that have previously adapted to the environment.

We model self–organizing systems as a set of units subject to change. The units are connected to and interact with each other through directed edges or links, therefore these systems can be called networks. An alliance is a distinguished set of units. In the dissertation, we use a different terminology for each alliance depending on the nature of the underlying framework. A unit may belong to more than one alliance. *The memory of the alliance* is the set of links between the members of the given alliance. *The work of the alliance* is characterized by the rules aiming at the optimization of rewards obtainable by the alliance. The work of the alliance may also include the reward sharing rules. The self–organizing property signifies that new dependencies may be formed during the work of the alliance on the basis of the interactions between the units and the rewards that the units collect. The system is selective provided that the reward sharing rules induce competition

among the units.

1.1.1 The Scale–Free Small World Phenomenon

The dissertation deals with self–organizing networks that compile to the scale–free small world model. Over the past few decades a plethora of discoveries have initiated a revival of network modelling, resulting in the introduction and study of three main classes of modelling paradigms: the random graph, the small world and the scale–free world modelling paradigms.

The theory of random graphs was introduced by Pál Erdős and Alfréd Rényi [39, 40, 41]. A detailed review of the field is available in the classic book of Bollobás [16]. Erdős and Rényi defined a random graph as N labelled nodes connected by n edges, which are chosen randomly from the possible edges [39, 40, 41]. The graphs that can be created form a probability space in which every realization is equiprobable. Despite the random placement of links, most nodes are assigned approximately the same number of links. As a consequence, rarely can a node be found that has a significantly greater or smaller number of links than a randomly chosen node. Erdős and Rényi discovered that a multitude of properties of random graphs appear quite suddenly. The Erdős–Rényi model, however, raises the question as to whether the networks observed in nature are random indeed or whether they obey some organizing principles encoded in their topology.

The small world concept describes the fact that in spite of their often large size, in most networks there is a relatively short path between any two nodes [3]. The most popular manifestation of small worlds is the *six degrees of separation* concept. Stanley Milgram concluded that there is a path of acquaintances with a typical length of about six between most pairs of people in the United States [69, 91, 127]. Erdős and Rényi showed that the typical distance between any two nodes in a random graph scales as the logarithm of the number of nodes. Thus random graphs are small worlds, as well. The existence of the small world effect had been pondered about before Milgram's work, in a short story of Frigyes Karinthy [66]. The phrase did not appear in Milgram's writing, it was coined some decades later by Guare [53]. Garfield gave a brief review of Milgram's experiment and works

stemming from it in [50]. In 1998, Duncan J. Watts and Steven Strogatz demonstrated that regular graphs (in which the diameter is proportional to the size of the network), by adding only a small number of long–range links to them, can be transformed into small worlds [129]. Over the past few years researchers discovered a broad range of networks that exhibit the *small world phenomenon*. Examples include the citation pattern in science, collaboration among actors in Hollywood playing in the same movie [1], the neural networks of C. elegans, power grids, social and economic disparities governing competitive systems [94] etc. The small world effect accounts for the spread of information or a rumour across a network.

Small world networks have high clustering coefficients [129]. This concept has roots in sociology, it also appears under the name *fraction of transitive triples* [128]. The clustering coefficient quantifies the closeness of a vertex and its neighbours to being a clique, i.e. a complete graph. Clustering serves for extracting *community structure* from a network [51, 94]. In fact, a group of vertices that have a high density of edges within them and lower density of edges with the outside world, i.e. vertices belonging to other groups, constitute a cluster. During the clustering process, the edges are removed iteratively. The removal of edges is based on the *connection strength*. For the different possibilities of defining the connection strength, consult [51, 94]. The algorithm finishes locally on condition that no local change occurs in the network structure. The algorithm terminates globally, in case the structure of the network is not modified at global level. The created components constitute communities or *clusters* [51, 94].

Networks with power–law degree distributions are referred to as scale–free networks [12, 13, 15]. The emergence of the power–law degree distribution can be traced back to two mechanisms. First, the number of nodes in the network increases over time. Secondly, nodes with a large number of connections are more likely to accumulate new links than those with only a few connections. Briefly, the network exhibits *preferential attachment*. The term *preferential attachment* was coined by Barabási and Albert in 1999 to describe network growth [12, 14], though, scientists had already employed it to characterize other phenomena of the same nature under various names. In 1925, Yule used it to explain the

[1] The small world concept has given rise to some well–known parlour games, particularly the calculation of Erdős numbers (see The Erdős Number Project at http://www.oakland.edu/enp) and Bacon numbers (see The Oracle of Bacon at http://OracleOfBacon.org).

power–law distribution of the number of species per genus of flowering plants [132]. In his honour, the process is sometimes called *Yule process*. Herbert Simon demonstrated that the power–law distribution arises when *the rich gets richer* [19, 117]. Simon claims that wealth is distributed among individuals according to how much they have already had. In sociology, this phenomenon is referred to as *Matthew effect* [90], after a passage of the Gospel of Matthew: "For unto every one that hath shall be given, and he shall have in abundance: but from him that hath not shall be taken away that which he hath." (Matthew 25:29, King James Version.) The preferential attachment process, however, does not include withdrawal. In 1965, Price applied preferential attachment to citation networks of scientific papers [118]. Price invented the name *cumulative advantage* to explicate the cause of the emergence of power–law distribution [119]. The power–law degree distributions may be observed in a host of networks, including the World Wide Web, the Internet, metabolic networks, telephone call graphs [94] etc. The structure of a scale–free network, unlike that of a random network, is inhomogeneous: in scale–free networks there are far more weakly connected nodes than nodes with a large number of links. The high degree nodes are often called *hubs*. The hubs guarantee that the system is fully connected. Scale–free networks cannot be destroyed by the removal of any finite fraction of nodes, since the random removal most likely affects the less–connected nodes, which does not damage seriously the network topology. On the other hand, scale–free networks are extremely vulnerable to targeted attacks [4, 11]. Indeed, the removal of a small fraction of the most connected nodes can cause the network to fall apart. Consequently, a hub can be considered both as a strength and as an Achilles' heel of a scale–free network.

1.2 Grammar Systems Theory

Self–organizing systems typically have higher level properties that cannot be observed at the level of the elements [24]. These properties can be seen as the product of the interactions of these elements. The arising of novel and coherent structures, patterns, and properties during the process of self–organization in complex systems are referred to as emergence. The emergent phenomena are conceptualized as occurring on a macro scale, in contrast to the micro–scale components and processes out of which they arise.

In grammar systems theory the functionality of an agent can be viewed as an emergent property of its intensive interaction with its dynamic environment [30]. If the choice of the agent is pre–wired, then it can be described as non–deliberative or reactive. It implies that the agent has no access to the change of the internal representation of the environment and that the knowledge is implicitly or explicitly embedded into the structure of the agent.

The theory of grammar systems regards formal languages as a set of sequences of symbols describing the behaviour of complex systems of cooperating and communicating agents at symbolic level [30, 110]. The grammars can be interpreted as agents, whilst the generated language describes the behaviour of the system. According to the traditional approach of formal language theory, a language is produced by a grammar or another language generating mechanism, whereas according to the unconventional approach of grammar systems theory a language may be the product of the cooperation or the joint action of several grammars or language generating mechanisms. The characteristics of a systems are determined through the individual and the collective behaviour of its members. In the sequel, we review the grammar systems theoretical constructions, in particular, networks of language processors and eco–grammar systems, which serve as a basis for describing the self–organizing networks that we will present in this dissertation.

1.2.1 Networks of Language Processors

Networks of language processors or NLP systems are a term coined for the description of various features of different architectures of parallel and distributed processing such as the WAVE architecture [42, 114], the Boltzmann machine [43] and the Connection Machine [57, 126], motivated by the data flow paradigm. One of the paradigms of data flow in parallel and distributed environment is the Logic Flow paradigm [42, 114], according to which processors are placed in the nodes of a virtual complete graph and able to handle data associated with the respective node. Data processing is commenced by the injection of some data in some node(s). Each node possessing data performs data processing locally under well–defined conditions. The data is transmitted to other nodes in the graph through replication or splitting. It is the patterns to be matched that define the target nodes of the data sent. The recipients may deal with the messages that they have received in different

manners. In the case of the Connection Machine, for instance, processors to which multiple messages have been sent either compute the bitwise logical *or*, the numerically largest or the integral sum of all the messages [126].

Parallel communicating grammar systems with communication by command (CCPC grammar systems) were the first models in the field of networks of language processors [32]. The communication is compulsory after each rewriting step at the level of the system, the language processors communicate by command. In CCPC grammar systems at the beginning of the computation only one word is present at each node. As the computation progresses, rewriting and communication steps alternate. During the rewriting step the component grammars continue the derivation until they have some production to be applied. In the course of the communication step messages belonging to the selector language of the recipient are concatenated. The selector language, in fact, is a regular filter. Only one entrance filter is associated with each node.

The general framework of networks of language processors, including systems communicating by command was presented in [34]. Networks of language processors (see, e.g. [27, 34]) consist of several language identifying devices or components associated with the nodes of a virtual graph. The language processors operate on strings by performing rewriting and communication steps alternately. A system of networks of language processors functions by changing its states. At any step, the state of the network is described by the sets of strings present at the components. During the rewriting step, some strings present at some component are rewritten according to the rewriting rule set and rewriting mode of the component. During the communication step, some strings, or copies of some strings present at some component and satisfying some context condition are communicated to other components via input and output filters. In other words, the communication is successful provided that the communicated string is able to penetrate the exit filter of the sender (satisfies the context condition given by the exit filter) and the entrance filter of the receiver (fulfills the context condition imposed by the entrance filter). Besides the filters, a communication graph may be associated with the network so that the target nodes can be prescribed by the communication protocol for the senders.

1.2.2 Eco–Grammar Systems

An eco–grammar system (an EG system) aims at modelling the interplay between the *environment* and the *agents* in complex systems such as ecosystems [26, 31]. The original model is based on the following postulates [31]:

1. An ecosystem consists of an environment and a set of agents. Both the state of the environment and the states of the agents are described by strings of symbols from given alphabets.

2. In an ecosystem there is a *universal clock* that marks the time units, according to which the evolution of the agents and of the environment occurs.

3. Both the environment and the agents have *developmental rules*[2], which are rules of Lindenmayer systems. These developmental rules are applied in a parallel manner to all the symbols describing the state of an agent and the state of the environment. A (rewriting) step is performed in each time unit.

4. The developmental rules of the environment are *independent* of the agents and of the state of the environment. The developmental rules of the agents, however, *depend* on the state of the environment (at a given moment, a subset of the rules is chosen from a general set associated with each agent).

5. The agents act on the environment (and possibly on other agents) by employing their context–free *action rules*. In each time unit, every agent utilizes one action rule chosen from a set depending on the current state of the agent.

6. The *action* (of agents on the environment) *has priority over the evolution* of the environment. In a given time unit, those symbols of the environment that are not affected by the actions of the agents are rewritten (in a parallel manner) by the developmental rules.

[2]In [31], the authors use evolution rules. In order to unify the terminology, we employ developmental rules in lieu of evolution rules.

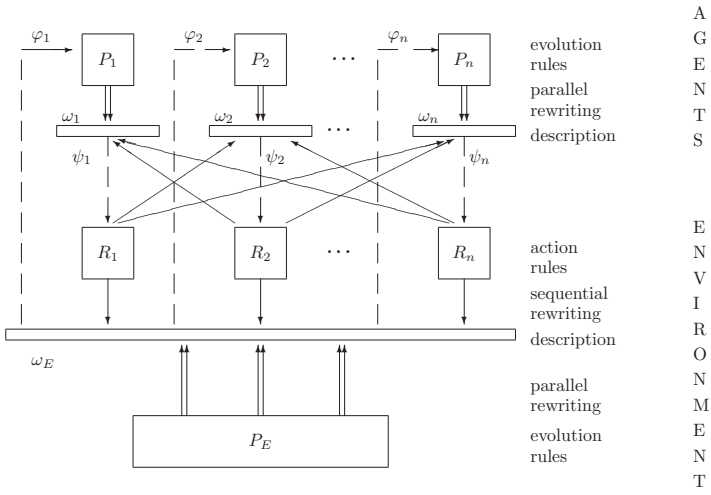

Figure 1.1: **Eco–grammar systems.** The general version as it was presented in [31].

The conceptual construction of an EG system is presented in Fig 1.1. We distinguish the environment and n agents, $A_1, \ldots, A_n$. The state of the environment is described by a string ω_E over some alphabet V_E and it develops according to a set P_E of interactionless Lindenmayer rules (0L rules). The state of each agent A_i is described by a string ω_i, over an alphabet V_i and it evolves according to a set P_i of 0L rules. At a given moment, only a subset $\varphi_i(\omega_E)$ of P_i is *active*, depending on the current state of the environment. Moreover, each agent A_i has an associated set R_i of rewriting rules by which A_i acts, locally, on the environment or on another agent. These rules are used in a sequential manner (using $x \to y$ means replacing exactly one occurrence of x by y). At a given moment, a subset $\psi_i(\omega_i)$ of R_i is *active*, depending on the current state of A_i. A rule $x \to y$, with x, y consisting of symbols from V_E, is used to perform an action on the environment, whereas a rule with x, y over an alphabet V_j, is applied to act on the j–th agent. The whole life of the system is supposed to be governed by a universal clock, dividing the time in unit intervals: in every time unit, the agents act on the environment or on other agents (using exactly one action

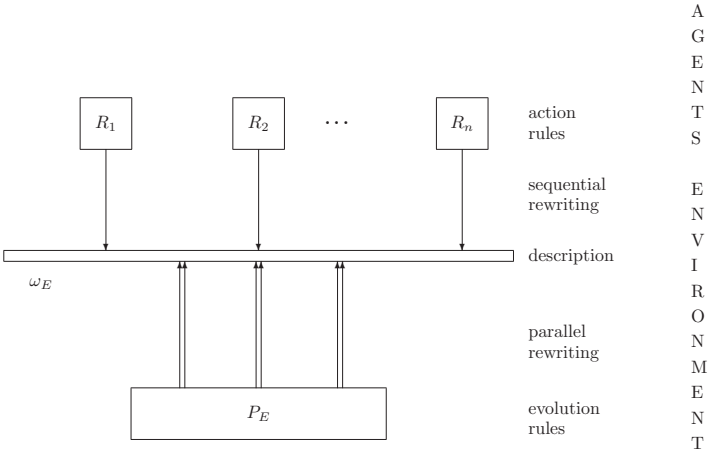

Figure 1.2: **Simple eco–grammar systems.** The simplification of the general model implies the omission of functions φ_i, ψ_i and of the inner representation of the agents.

rule), then the developmental rules rewrite, in parallel, all the remaining symbols in the strings describing the environment and the agents. Thus the action has priority over the evolution.

In the original model of eco–grammar systems both the environment and the agents have inner representation and the interaction between the different components of the system is rather complex [26, 31]. The complex nature of the evolution of eco–grammar systems deduced on the basis of their life–cycle and the attainable derivation sequences as well as the appropriate choice of evolution functions φ_i and ψ_i in order to generate all recursively enumerable languages imposed some restrictions on the general model. As a consequence, the notion of simple eco–grammar systems (SEG systems) was introduced [31]. In SEG systems, the agent do not evolve, neither do their actions on the environment depend on the state of the system. The conceptual construction of a SEG system is illustrated in Fig 1.2.

1.3 Aims of the Research

In this dissertation we give a formal language theoretic approach to self–organizing networks. Not only do we model self–organizing networks at syntactical level, but we also reveal some semantical and experimental aspects related to them. In effect, we argue the applicability of syntactical devices to a wide range of phenomena that may arise within the realm of these networks.

First, we deal with peers forming ad–hoc communities, i.e. peer groups in peer–to–peer networks, based on their interest and expertise. To capture the behaviour of peer–to–peer networks, we modify the definition of networks of parallel multiset string processors with teams of collective filtering: we equip the peers with the individual filtering mechanism. We establish the connection between the growth of the number of strings being present during the computation at the components of these networks of parallel multiset string processors and the growth functions of certain types of developmental systems (Lindenmayer systems). We demonstrate how the formal language theoretic model can be employed to incorporate network security requirements. Protection mechanisms are needed to defend communities against malicious actions of outsiders. Security constraints ought to be guaranteed at intra– and intercommunity levels. Our main focus is to elaborate a flexible, dynamic access control model. Instead of defining global access control model to which each local sub–system must conform, we allow each sub–system to define their own access control requirements. Global consistency is achieved by the recursive refinements of the local requirements. Our results indicate that our approach is promising from both the perspectives of expressiveness as well as efficiency. In particular, our approach allows quick and efficient local analysis of security requirements, thus reducing the need for global verification.

Secondly, we present an approach to the behaviour of the complex system of cooperating and communicating Internet crawlers seeking novel information on the World Wide Web. We apply eco–grammar systems in which the agents are represented by regulated rewriting devices to describe the information harvest of the crawlers. These generative mechanisms impose some constraint on the search strategy of the agent. We verify that if we ignore the aging of the web pages in the model, then systems with rather simple component grammars suffice to the class of recursively enumerable languages. Whereas if the web

pages may become obsolete, then the efficiency of the cooperation of the agents decreases considerably. Our aim is to illustrate the great diversity of employing regulated rewriting devices in the field of web crawling techniques.

We use simulations to study the behaviour of our model crawlers. In these simulations, we utilize the methods of selective learning and value estimation under *evolutionary pressure* [23], [49]. We compare the selective learning algorithm to the linear function approximation–based reinforcement learning algorithm [122]. The selection is influenced by the ever changing external world and by the competing individuals. Communication can either occur directly or indirectly. The indirect communication is invoked by the reward system: positive reward is delivered only to the first sender of a news item. The crawlers have to find either novel documents, novel documents within time limits, novel documents on different topics or documents satisfying some possible combination of the previous criteria. We investigate the extent to which communication makes a goal–oriented community efficient in different graph topologies. We conduct experiments in scale–free small worlds and some of our theoretical results are also related to scale–free small worlds, since our algorithms prove to be efficient in these structures. Furthermore, we examine the efficiency of the algorithms in scale–free networks not fulfilling the small world condition and in random networks, as well.

Finally, we model the behaviour of agents participating in network cluster formation based on local means. In networks characterized by the small world phenomenon and by high clustering coefficients, information propagation occurs in a highly efficient manner. The communicating and collaborating agents create a hierarchical network structure. We extend the conditions of dynamic team constitution in eco–grammar systems. Through their actions, the agents contribute to the solution of various tasks. From the language classes that these systems are capable of generating, we deduce the difficulty of the problem they can solve given the various team constitution modes.

1.4 Outline of the Dissertation

Mirroring the themes introduced above, the dissertation is divided into a number of parts. In Chapter 2, we overview the mathematical definitions and theorems employed throughout the dissertation. From there we move to the foundational modelling ideas that are the focus of our activity. In particular, we deal with peer–to–peer networks, Internet crawlers in quest of novel information and network cluster formation based on local means in Chapter 3, Chapter 4 and Chapter 5, respectively. Finally, we conclude our dissertation with a summary in English and in Hungarian of the contributions of our work to the field of self–organizing networks.

Chapter 2

Preliminaries

In this chapter we review the basic concepts from the points of view of self–organizing networks and of formal language theory. For the sake of legibility, only the most important notions used throughout this dissertation are revised. For a finite set A, $card(A)$ stands for the number of elements of A. The set of natural numbers is denoted by $\mathbb{N}$ and $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$.

2.1 Prerequisites Related to Self–Organizing Networks

In this section we present the concepts related to the self–organizing systems: network models and reinforcement learning methods.

2.1.1 Network Models

Herein we give a brief overview of the network modelling paradigms to which we refer in this dissertation. For other aspects of network modelling, see, e.g. [94].

Random Graphs

Erdős and Rényi described random graphs in [39, 40, 41]. Let $E_{N,n}$ denote the set of all graphs having N, $N \geq 1$, given labelled vertices $V_1, V_2, \ldots, V_N$ and n, $n \geq 1$, edges. The graphs considered are supposed to be undirected. A graph belonging to the set $E_{N,n}$ is obtained by choosing n out of the $\binom{N}{2}$ possible edges connecting points $V_1, V_2, \ldots, V_N$, and therefore the number of elements of $E_{N,n}$ is equal to $\binom{\binom{N}{2}}{n}$. A random graph $\Gamma_{N,n}$ can be defined as an element of $E_{N,n}$, chosen at random so that each element of $E_{N,n}$ has the same probability to be chosen, namely $\frac{1}{\binom{\binom{N}{2}}{n}}$. We may consider the formation of a random graph as a stochastic process, as well. At time $t = 1$, we choose one out of the $\binom{N}{2}$ possible edges connecting points $V_1, V_2, \ldots, V_N$, each of these edges having the same probability to be chosen; let this edge be denoted by e_1. At time $t = 2$, we choose one of the possible $\binom{N}{2} - 1$ edges, different from e_1, all these being equiprobable. Continuing this process at time $t = k + 1$, we choose one of the $\binom{N}{2} - k$ possible edges different from edges $e_1, e_2, \ldots, e_k$ already chosen, each of the remaining edges being equiprobable, i.e. having the probability $\frac{1}{\binom{N}{2}-k}$. We denote by $\Gamma_{N,n}$ the graph consisting of vertices $V_1, V_2, \ldots, V_N$ and edges $e_1, e_2, \ldots, e_N$. The two definitions are equivalent [41].

Watts–Strogatz Model

Watts and Strogatz proposed a one–parameter model that interpolates between an ordered finite dimensional lattice and a random graph [129].

The algorithm that generates the model is as follows:

1. We start with a ring of N vertices, each connected to its K nearest neighbours ($K/2$ on either side) by undirected edges. We choose a vertex and the edge that connects it to its nearest neighbour in the clockwise sense. With probability p, we reconnect this edge to a vertex chosen uniformly at random over the entire ring, with duplicate edges excluded, otherwise we leave the edge in place. We repeat this process by moving clockwise around the ring, considering each vertex in turn until one lap is completed.

2. At the next step, we consider the edges that connect vertices to their second nearest neighbours clockwise. As before, we randomly rewire each of these edges with probability p, and continue this process, circulating around the ring and proceeding outward to more distant neighbours after each lap, until each edge in the original lattice has been considered once. (As there are $NK/2$ edges in the entire graph, the rewiring process stops after $K/2$ laps.)

This construction allows us to *tune* the graph between regularity ($p = 0$) and disorder ($p = 1$), and thereby to probe the intermediate region $0 \leq p \leq 1$, about which little is known.

We quantify the structural properties of the graphs obtained by the random rewiring procedure of the Watts–Strogatz model by their characteristic path length $L(p)$ and clustering coefficient $C(p)$. In effect, $L(p)$ measures the typical separation between two vertices in the graph (a global property), whereas $C(p)$ measures the cliquishness of a typical neighbourhood (a local property). The clustering coefficient $C(p)$ is defined as follows. Suppose that a vertex v has k_v neighbours, then at most $\frac{k_v(k_v-1)}{2}$ edges can exist between them (this occurs when every neighbour of v is connected to every other neighbour of v). The ratio between the number of edges that actually exist between these k_v nodes and the total number $\frac{k_v(k_v-1)}{2}$ gives the value of the clustering coefficient of node v, C_v. The clustering coefficient of the whole network is the average of C_v over all v.

In order to have a sparse but connected network at all times, $N \gg K \gg \ln(N) \gg 1$ is required. Watts and Strogatz observed that $L \sim N/2K \gg 1$ and $C \sim 3/4$ as $p \to 1$, whilst $L \approx L_{random} \sim \ln(N)/\ln(K)$ and $C \approx C_{random} \sim K/N \ll 1$ as $p \to 0$. Consequently, the regular lattice at $p = 0$ is a highly clustered, large world, where L grows linearly with N. On the other hand, the random network at $p = 1$ is a poorly clustered, small world, where L grows only logarithmically with N.

The random rewiring procedure of the Watts–Strogatz model is illustrated in Fig. 2.1.

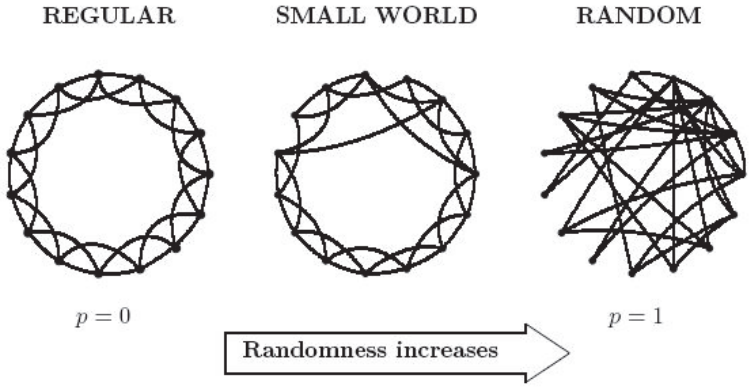

Figure 2.1: **The random rewiring procedure of the Watts–Strogatz model.** Starting from a ring lattice with N vertices and K edges per vertex, we rewire each edge at random with probability p. The realization of the algorithm is shown, for different values of p. For $p = 0$, the original ring is unchanged; as p increases, the graph becomes increasingly disordered until for $p = 1$, all edges are rewired randomly. Watts and Strogatz demonstrated that for intermediate values of p, the graph is a small world network: highly clustered like a regular graph, yet with small characteristic path length, like a random graph [129].

Scale–Free Networks

Networks with degree distributions decaying as a power–law are called scale–free networks [12, 13, 15]. In other words, the probability $P(k)$ that a vertex in the network interacts with nodes in the network having k connections to other nodes goes for large values of k as $P(k) \sim k^{-\gamma}$, where γ is a constant with a value typically in the range $2 < \gamma < 3$, although occasionally this value may lie outside these bounds.

Barabási–Albert Model

The Barabási–Albert model is the result of an algorithm that generates random scale–free networks using the preferential attachment mechanism [12, 13, 15]. Let us suppose that the initial network has m_0 nodes, where $m_0 \geq 2$. Furthermore, let us assume that in the beginning the degree of each node is at least 1, otherwise the network will always remain disconnected. At every time step, we add a new vertex to the network with m ($\leq m_0$) edges linking the new vertex to m different vertices already present in the system. The probability Π that a new vertex will be connected to vertex i depends on the connectivity k_i of that vertex, such that $\Pi(k_i) = \frac{k_i}{\sum_j k_j}$.

2.1.2 Reinforcement Learning

Reinforcement learning (RL) is an approach to sequential decision making in an unknown environment through learning from past interactions [122]. The learner and decision–maker is called the agent. The agent interacts with the environment, i.e. it selects actions. In most cases, RL problems are treated as Markov decision processes (MDPs), i.e. states are fully observable and rewards and successor states depend only on the current state and action but not on the history. An MDP is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}^a_{ss'}, \mathcal{R}^a_{ss'}, \gamma)$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $\mathcal{P}^a_{ss'}$ is the probability of reaching state s' after taking action a in state s, $\mathcal{R}^a_{ss'}$ is the reward received when the transition occurs and $0 \leq \gamma \leq 1$ is the discount rate parameter. A trajectory of experience is a sequence $s_1, a_1, r_2, s_2, a_2, r_3, s_3, \ldots$, where the agent in s_1 takes action a_1 and receives reward r_2 while making a transition to s_2 etc. At each time step, the agent employs a mapping $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ from each state $s \in \mathcal{S}$ and action $a \in \mathcal{A}(s)$ ($\mathcal{A}(s)$ denotes the fact that the selected action a depends on s) to the probability $\pi(s, a)$ of taking action a in state s. This mapping is called the *policy* of the agent. Let us denote the *value* of a state s under a policy π by $V^\pi(s)$, which is the expected sum of discounted future rewards when starting in s and following π thereafter. For MDPs, we can define $V^\pi(s)$ formally as:

$$V^{\pi}(s) = E_{\pi}[\sum_{t=1}^{\infty} \gamma^{t-1} r_t \mid s_0 = s, \pi],$$

where $E_{\pi}[.]$ denotes the expected value given that the agent follows policy π, s_0 is the initial state, r_t, is the immediate reward at time t, $t \geq 0$, and γ, $0 \leq \gamma \leq 1$, is the discount rate parameter.

Denoting by s_t the state the agent is in at time t, $t \geq 0$, we can write the value function recursively as:

$$\begin{aligned} V^{\pi}(s) &= \sum_{a} \pi(s,a) \sum_{s'} \mathcal{P}^{a}_{ss'} (\mathcal{R}^{a}_{ss'} + \gamma V^{\pi}(s')) \\ &= E_{\pi}[r_{t+1} + \gamma V^{\pi}(s_{t+1}) \mid s_t = s, \pi]. \end{aligned}$$

The agent's goal, roughly speaking, is to maximize the total amount of reward it receives over the long run.

For a particular value function V, we define the TD [1] error δ_t at time t, $t \geq 0$, as:

$$\delta_t(V) = r_{t+1} + \gamma V(s_{t+1}) - V(s_t),$$

where s_t is the state the agent is in at time t, $t \geq 0$, s_{t+1} is the state that the agent chooses at the next step and r_{t+1} is the reward to be collected in s_{t+1}.

2.2 Formal Language Prerequisites

The reader is assumed to be familiar with the basics of formal language theory, for further details consult [36, 109, 110, 112].

[1] The abbreviation TD stands for *temporal differencing*. For a detailed description of TD methods, consult e.g. [122]

For an alphabet V, we denote by V^* the set of words over V, by V^+ the set of all nonempty words, i.e. $V^+ = V^* \setminus \{\lambda\}$, where λ is the empty string. Let $(x)_{V'}$ and $|x|_{V'}$ be the string and the length of the string, respectively, obtained through the erase of the symbols that are not in $V' \subseteq V$. If $V' = \{a\}$, we simply write $(x)_a$ and $|x|_a$. Let $length(x)$ denote the length of $x \in V^*$ and $alph(x)$ the set of symbols occurring in $x \in V^*$. Moreover, for $L \subseteq V^*$, let $alph(L) = \bigcup_{x \in L} alph(x)$.

Let U denote the set (the universe) of objects. A multiset is a pair $M = (V, f)$, where V is an arbitrary (not necessarily finite) set of objects of U and $f : U \to \mathbb{N}_0$ is a mapping assigning the multiplicity to each object, such that if $a \notin V$, then $f(a) = 0$. The support of $M = (V, f)$ is defined by $supp(M) = \{a \in V \mid f(a) \geq 1\}$. M is a finite multiset in case $supp(M)$ is finite. The set of all finite multisets over the set V is denoted by V°. The number of objects in a finite multiset $M = (V, f)$, or in other words, the cardinality of M, is defined by $card(M) = \sum_{a \in V} f(a)$. For instance, a multiset with elements a, a, a, b, b, c is denoted by $\{\{a, a, a, b, b, c\}\}$. We claim that $a \in M = (V, f)$, if $a \in supp(M)$, and $M_1 = (V_1, f_1) \subseteq M_2 = (V_2, f_2)$, if $supp(M_1) \subseteq supp(M_2)$ and for all $a \in V_1, f_1(a) \leq f_2(a)$. We define the union of two multisets by $(M_1 \cup M_2) = (V_1 \cup V_2, f')$, where for all $a \in V_1 \cup V_2, f'(a) = f_1(a) + f_2(a)$. M is an empty multiset, denoted by ϵ, provided that $supp(M) = \emptyset$. A multiset M over the finite set of objects V can be represented as a string ω over the alphabet V with $|w|_a = f(a)$, $a \in V$ and λ representing the empty multiset ϵ. In the sequel, the finite multiset of objects with the word ω over V representing M is identified by $M = (V, f)$, hence $\omega \in V^\circ$ is written.

By a context condition ϱ over V^*, where V is an alphabet, we mean a computable mapping $\varrho : V^* \to \{true, false\}$. We say that ϱ is of type

1. **reg**, or it is a regular context condition over V^*, given by a regular language $L \subseteq V^*$, if $\varrho(\omega) = true$ for any $\omega \in V^*$, where $\omega \in L$, otherwise $\varrho(\omega) = false$.

2. **rc**, or it is a random context condition over V^*, given by a pair (Q, R), where $Q, R \subseteq V$, if $\varrho(\omega) = true$ for any $\omega \in V^*$ that contains each element of Q, but no element of R and $\varrho(\omega) = false$ otherwise. By definition, Q and R can be empty sets, in this case we omit the corresponding context check. Q is called the permitting and R the forbidding context condition.

A *phrase structure grammar* is a quadruple $G = (N, T, S, P)$, where N is the nonterminal alphabet, T is the terminal alphabet, $N \cap T = \emptyset$, $S \in N$, is the start symbol or axiom and P is a (finite) set of rewriting rules. We say that $x \in (N \cup T)^*$ directly derives $y \in (N \cup T)^*$, written as $x \Longrightarrow y$ iff $x = x_1 u x_2, y = x_1 v x_2$, for some $u \to v \in P$. The reflexive and transitive closure of $\Longrightarrow$ is denoted by $\Longrightarrow^*$. The language generated by G is defined by $L(G) = \{x \in T^* \mid S \Longrightarrow^* x\}$.

A phrase structure grammar $G = (N, T, S, P)$ is context–sensitive, if each production $u \to v \in P$ has $u = u_1 A u_2$, $v = v_1 x v_2$, for $u_1, u_2 \in (N \cup T)^*$, $A \in N$, $x \in (N \cup T)^+$ and if $S \to \lambda \in P$, then S does not occur on right side of any productions. A phrase structure grammar G is context–free, if each $u \to v \in P$ has $u \in N$. A phrase structure grammar G is regular, if each $u \to v \in P$ has $u \in N$ and $v \in T \cup TN \cup \lambda$.

The families of languages generated by regular, context–free, context–sensitive and phrase structure grammars are denoted by $\mathcal{L}(\mathrm{REG})$, $\mathcal{L}(\mathrm{CF})$, $\mathcal{L}(\mathrm{CS})$ and $\mathcal{L}(\mathrm{RE})$, respectively.

Let G be a grammar of arbitrary type and let N, T and S be its nonterminal alphabet, terminal alphabet and start symbol, respectively. For a derivation $D : S = \omega_1 \Rightarrow \omega_2 \Rightarrow \ldots \Rightarrow \omega_r = \omega \in T^*$ according to G, we set $Ind(D, G) = max\{|\omega_i|_N \mid 1 \leq i \leq r\}$, and for $\omega \in T^*$, we define $Ind(\omega, G) = min\{Ind(D, G) \mid D \text{ is a derivation for } \omega \text{ in } G\}$. The *index of grammar* G is defined as $Ind(G) = sup\{Ind(\omega, G) \mid \omega \in L(G)\}$. For a language L in the family $\mathcal{L}(X)$ of languages generated by grammars of some type X, we define $Ind_X(L) = inf\{Ind(G) \mid L(G) = L, G \text{ is of type } X\}$. If no confusion arises, we write $Ind(L)$ instead of $Ind_X(L)$. For a family $\mathcal{L}(X)$, we set $\mathcal{L}_n(X) = \{L \mid L \in \mathcal{L}(X) \text{ and } Ind_X(L) \leq n\}, n \geq 1$, and $\mathcal{L}_{fin}(X) = \bigcup_{n \geq 1} L_n(X)$.

A *pure context–free grammar* is a pair $\gamma = (V, \omega, P)$, where V is an alphabet, $\omega \in V^+$ is the axiom of the grammar and P is a finite set of rewriting rules of the form $a \to v$, where $a \in V$, $v \in V^*$. For two strings x and y in V^*, we say that x yields y in a direct derivation step by using a rule in P, written as $x \Rightarrow_P y$, if $x = x_1 a x_2$, $y = x_1 v x_2$, where $x_1, x_2 \in V^*$ and $a \to v \in P$. P is *complete*, if for each $a \in V$, there exists a rule $a \to x$ in P.

2.2.1 Grammars with Controlled Derivations

A *programmed grammar* with *appearance checking* is a construction $G = (N, T, S, P)$, where N, T are disjoint alphabets, $S \in N$, and P is a finite set of triplets or rules of the form $(l : A \to x, \sigma(l), \varphi(l))$, where $A \in N$, $x \in (N \cup T)^*$, $l \in Label(P), \sigma(l), \varphi(l) \subseteq Label(P)$, and $Label(P)$ is a set of labels associated with the triplets of P in a one–to–one manner. If only (N, S, P) is indicated, then we speak of a *programmed grammar scheme*.

For $(l : A \to x, \sigma(l), \varphi(l)) \in P$, we define $(\omega, l) \Rightarrow (\omega', h)$, iff either $\omega = \omega_1 A \omega_2, \omega' = \omega_1 x \omega_2, h \in \sigma(l)$, or A does not appear in $\omega, \omega = \omega'$ and $h \in \varphi(l)$, where $\sigma(l)$ is called the success and $\varphi(l)$ the failure field of the rule. If $\varphi(l) = \emptyset$, then G is a programmed grammar without appearance checking. The generated language is $L(G) = \{\omega \in T^* \mid (S, l_0) \Rightarrow (\omega_1, l_1) \Rightarrow \ldots \Rightarrow (\omega_m, l_m) = (\omega, l_m), l_i \in Label(P), \text{for } 0 \leq i \leq m\}$.

We denote by $\mathcal{L}(\mathrm{PR}_{ac})$ and $\mathcal{L}(\mathrm{PR}^{\lambda}_{ac})$ the families of languages generated by programmed grammars in the appearance checking mode with λ–free context–free rules and with arbitrary context–free rules, respectively. If the appearance checking feature is not present, then subscript *ac* is omitted.

A *matrix grammar* with *appearance checking* is a construction $G = (N, T, S, M, \mathcal{F})$, where N, T are disjoint alphabets, $S \in N$, and $M = \{m_1, m_2, \ldots, m_r\}$ is a finite set of sequences, called matrices, of the form $m_i : (A_{i_1} \to x_{i_1}, \ldots, A_{i_{k_i}} \to x_{i_{k_i}})$, where $A_{i_j} \in N$, $x_{i_j} \in (N \cup T)^*$, $1 \leq i \leq r, 1 \leq j \leq k_i$, and $\mathcal{F}$ is a set of occurrences of rules in the sequences of M.

For $m_i : (A_{i_1} \to x_{i_1}, \ldots, A_{i_{k_i}} \to x_{i_{k_i}}) \in M$, $1 \leq i \leq r, k_i \geq 1$, $\omega, \omega' \in (N \cup T)^*$, we define $\omega \Rightarrow_{m_i} \omega'$, iff there are $\omega_{i_1}, \ldots, \omega_{i_{k_i+1}} \in (N \cup T)^*$, such that $\omega = \omega_{i_1}, \omega' = \omega_{i_{k_i+1}}$ and for each $i, j, 1 \leq i \leq r, 1 \leq j \leq k_i$, either $\omega_{i_j} = \omega'_{i_j} A_{i_j} \omega''_{i_j}$ and $\omega_{i_{j+1}} = \omega'_{i_j} x_{i_j} \omega''_{i_j}$, or A_{i_j} does not occur in $\omega_{i_j}, \omega_{i_j} = \omega_{i_{j+1}}$ and $A_{i_j} \to x_{i_j}$ is an element of $\mathcal{F}$. If $\mathcal{F} = \emptyset$, then G is a matrix grammar without appearance checking. In this case the component $\mathcal{F}$ is omitted. The language generated by G is defined by $L(G) = \{\omega \in T^* \mid S \Longrightarrow_{m_{j_1}} y_1 \Longrightarrow_{m_{j_2}} y_2 \Longrightarrow_{m_{j_3}} \ldots \Longrightarrow_{m_{j_s}} \omega, 1 \leq j_i \leq r, 1 \leq i \leq s\}$.

We denote by $\mathcal{L}(\mathrm{MAT}_{ac})$ and by $\mathcal{L}(\mathrm{MAT}^{\lambda}_{ac})$ the families of languages generated by matrix

grammars in the appearance checking mode with λ–free context–free rules and with arbitrary context–free rules, respectively. When the appearance checking feature is not present, then subscript *ac* is left out.

A *random context grammar* with *appearance checking* is a construction $G = (N, T, S, P)$, where N, T are disjoint alphabets, $S \in N$, and P is a finite set of triplets or rules of the form $(A \to \omega, Q, R)$, where $A \in N$, $\omega \in (N \cup T)^*$, Q and R are subsets of N. For $x, y \in (N \cup T)^*$, we write $x \Longrightarrow y$, iff $x = x'Ax'', y = x'\omega x''$, for some $x', x'' \in (N \cup T)^*$, $(A \to \omega, Q, R)$ is a triplet in P, such that all symbols of Q appear in x', x'', and no symbols of R occur in x', x''. Q is called the *permitting*, R the *forbidding* context of rule $(A \to \omega, Q, R)$. If $R = \emptyset$, then G is a random context grammar without appearance checking. The language generated by a random context grammar G is defined by $L(G) = \{\omega \in T^* \mid S \Longrightarrow^* \omega\}$.

We denote by $\mathcal{L}(\mathrm{RC}_{ac})$ and by $\mathcal{L}(\mathrm{RC}^{\lambda}_{ac})$ the families of languages generated by random context grammars in the appearance checking mode with λ–free context–free rules and with arbitrary context–free rules, respectively. If the appearance checking feature is not present, then subscript *ac* is omitted.

It is known from [36] that

$$\mathcal{L}(\mathrm{CF}) \subset \mathcal{L}(\mathrm{PR}_{ac}) = \mathcal{L}(\mathrm{MAT}_{ac}) = \mathcal{L}(\mathrm{RC}_{ac}) \subset \mathcal{L}(\mathrm{CS}) \text{ and}$$
$$\mathcal{L}(\mathrm{PR}^{\lambda}_{ac}) = \mathcal{L}(\mathrm{MAT}^{\lambda}_{ac}) = \mathcal{L}(\mathrm{RC}^{\lambda}_{ac}) = \mathcal{L}(\mathrm{RE}).$$

A context–free matrix grammar $G = (N, T, S, M, \mathcal{F})$ is in the (preliminary) *2–normal form* iff

$$N = \{S\} \cup N^{(1)} \cup N^{(2)} \text{ with } N^{(1)} \cap N^{(2)} = \emptyset, S \notin N^{(1)} \cup N^{(2)},$$

and M contains only matrices of the following forms:

1. $S \to AX, A \in N^{(1)}, X \in N^{(2)}$,
2. $(A \to \beta, X \to Y), A \in N^{(1)}, \beta \in (N^{(1)} \cup T)^*, X, Y \in N^{(2)}$, and

3. $(A \to \beta, X \to \lambda), A \in N^{(1)}, \beta \in (N^{(1)} \cup T)^*, X \in N^{(2)}$.

Moreover, set $\mathcal{F}$ contains only rules of the form $A \to \beta$ in the matrices of types (2) and (3).

For each context–free matrix grammar G, an equivalent context–free matrix grammar G' can be constructed in the (preliminary) 2–normal form [36].

A *unordered scattered context grammar* is a construct $G = (N, T, S, P)$, where N, T are disjoint alphabets, $S \in N$, and $P = \{p_1, p_2, \ldots, p_r\}$ is a finite set of sequences of the form $p_i : (A_{i_1} \to x_{i_1}, \ldots, A_{i_{k_i}} \to x_{i_{k_i}})$, where $A_{i_j} \in N$, $x_{i_j} \in (N \cup T)^*$, $1 \leq i \leq r, 1 \leq j \leq k_i$. We say that ω directly derives ω', written as $\omega \Rightarrow \omega'$, iff for some i, $1 \leq i \leq r$, and for some permutation π of $\{1, \ldots, k_i\}$, $\omega = \omega_1 A_{i,\pi(j_1)} \omega_2 A_{i,\pi(j_2)} \ldots \omega_n A_{i,\pi(j_m)} \omega_{m+1}$, $\omega_j \in (N \cup T)^*$, for $1 \leq j \leq m+1$, $\omega' = \omega_1 x_{i,\pi(j_1)} \omega_2 x_{i,\pi(j_2)} \ldots \omega_n x_{i,\pi(j_m)} \omega_{m+1}$, and $\omega_1 \omega_2 \ldots \omega_{m+1}$ does not contain as subwords $x_{i,\pi(k)}$, $k \notin \{j_1, j_2, \ldots, j_m\}$. The generated language is $L(G) = \{x \in T^* \mid S \Longrightarrow^* x\}$.

We denote by $\mathcal{L}$(USC) and $\mathcal{L}_{fin}$(USC) the language families generated by unordered scattered context grammars and by unordered scattered context grammars of finite index, respectively.

2.2.2 Lindenmayer Systems

A 0L system (an *interactionless Lindenmayer system*) is a triplet $G = (V, \omega, P)$, where V is an alphabet, $\omega \in V^+$ is the axiom, and P is a finite set of context–free rewriting rules over V, such that for each $a \in V$, there is a rule $a \to x$ in P (we say that P is *complete*). For $z_1, z_2 \in V^*$, we write $z_1 \Longrightarrow z_2$ (with respect to G, if it is necessary, denoted by $\Longrightarrow_G$), if $z_1 = a_1 a_2 \ldots a_r, z_2 = x_1 x_2 \ldots x_r$, for $a_i \to x_i$ in P, $1 \leq i \leq r$. This form of derivation is called a 0L rewriting. The language generated by G is $L(G) = \{z \in V^* \mid \omega \Longrightarrow^* z\}$, where $\Longrightarrow^*$ is the reflexive and transitive closure of $\Longrightarrow$. We note that the reader may find notation $\Longrightarrow_P$ instead of notation $\Longrightarrow_G$ in the literature. The language family generated by 0L systems is denoted by $\mathcal{L}$(0L).

If for each $a \in V$, there is exactly one production of the form $a \to x, x \in V^*$, then we speak of a deterministic 0L, or a D0L system. If the axiom is replaced by a finite language, then we have a 0L (D0L) system with a finite number of axioms, or in other words, an F0L (FD0L) system.

Since the production set P of a D0L system $G = (V, \omega, P)$ defines a homomorphism $h : V \to V^*$, $G = (V, \omega, h)$ is often used instead of the first notation.

By a word sequence of a D0L system $G = (V, \omega, h)$, we mean the following sequence: $h^0(\omega) = \omega, h(\omega), h^2(\omega), h^3(\omega), \ldots$. Function $f : \mathbb{N}_0 \to \mathbb{N}_0$ defined by $f(t) = length(h^t(\omega))$, $t \geq 0$, is called the growth function of G, and sequence $length(h^t(\omega))$ for $t = 0, 1, 2, \ldots$, is its growth sequence.

A T0L system is a construct $G = (V, \omega, P_1, \ldots, P_n)$, $n \geq 1$, where each $G_i = (V, \omega, P_i)$, $1 \leq i \leq n$, is a 0L system. The language determined by G is $L(G) = \{z \in V^* \mid \omega \Longrightarrow_{G_{i_1}} \omega_1 \Longrightarrow_{G_{i_2}} \ldots \Longrightarrow_{G_{i_m}} \omega_m = z, 1 \leq i_j \leq n, 1 \leq j \leq m\}$. As above, in lieu of $\Longrightarrow_{G_{i_j}}$ we may write $\Longrightarrow_{P_{i_j}}$, $1 \leq i_j \leq n, 1 \leq j \leq m$. The language family generated by T0L systems is denoted by $\mathcal{L}$(T0L).

2.2.3 Networks of Parallel Language Processors

In the sequel, we give a brief review of networks of parallel language processors or NPLP systems. NPLP systems are particular cases of networks of language processors in which the components are represented by Lindenmayer systems. For the sake of legibility, we consider the case when the nodes are F0L systems, i.e. 0L systems with a finite set of axioms.

Definition 1 *A network of parallel language processors with* F0L *components or an* NPLP_{F0L} *system of degree* n, $n \geq 1$, *is a construct*

$$\Gamma = (V, (P_1, F_1, \varrho_1, \sigma_1), \ldots, (P_n, F_n, \varrho_n, \sigma_n)),$$

where

- *V is an alphabet, the alphabet of the system,*
- *$(P_i, F_i, \varrho_i, \sigma_i)$, $1 \leq i \leq n$, is called a component (a node) of the system (the i–th component or the i–th node), where*
  - *P_i, $1 \leq i \leq n$, is a finite and complete set of pure context–free rules over V (i.e. rules of the form $A \to \alpha$ with $A \in V$, $\alpha \in V^*$, and for each $A \in V$, there is a rule $A \to \alpha$ in P_i), the production set of the component,*
  - *$F_i \subset V^*$, $1 \leq i \leq n$, is a finite set, the set of axioms of the component,*
  - *ϱ_i, σ_i, $1 \leq i \leq n$, are context conditions over V^*, called an exit filter and an entrance filter of the component.*

According to the type of context conditions ϱ_i and σ_i, $1 \leq i \leq n$, we distinguish regular (reg) and random context (rc) exit and/or entrance filters. An NPLP_{F0L} system employs a homogeneous filtering mechanism of type (X), where $X \in \{reg, rc\}$, iff all the exit and entrance filters associated with the components of the network are of type (X). If the latter condition is not fulfilled, then the filtering mechanism is hybrid [73]. Should each P_i, $1 \leq i \leq n$, be a finite set of D0L rules or tables of a T0L system, then the underlying system is called an $\text{NPLP}_{\text{FD0L}}$ system or an $\text{NPLP}_{\text{FT0L}}$ system, respectively. If each axiom set consists of a single word, then letter F may be omitted from the notation. An NPLP system functions through state (configuration) transmissions.

Definition 2 *A state (or a configuration) of an* NPLP_{F0L} *system* $\Gamma = (V, (P_1, F_1, \varrho_1, \sigma_1), \ldots, (P_n, F_n, \varrho_n, \sigma_n))$, $n \geq 1$, *is a tuple* $s = (L_1, \ldots, L_n)$, *where* $L_i \subseteq V^*$, $1 \leq i \leq n$. $s_0 = (L_1, \ldots, L_n)$ *is called the initial state (initial configuration) of the system.*

A state (configuration) is changed either by a rewriting or a communication step. If a rewriting step occurs, then from each string at the disposal of a node only one new string can be derived through the application of 0L productions. In the course of a communication step, all nodes receive a copy of the strings present at the sender node provided that the string is able to penetrate the exit filter of the sender and the entrance filter of the receiver, i.e. it satisfies the corresponding context conditions. Rewriting and communication steps follow each other alternately.

Definition 3 *Let* $\Gamma = (V, (P_1, F_1, \varrho_1, \sigma_1), \ldots, (P_n, F_n, \varrho_n, \sigma_n))$, $n \geq 1$, *an* $\mathrm{NPLP}_{\mathrm{F0L}}$ *system,* $s_1 = (L_1, \ldots, L_n)$ *and* $s_2 = (L'_1, \ldots, L'_n)$ *two states (configurations) of* Γ, *where* $L_i, L'_i \subseteq V^*$, $1 \leq i \leq n$. s_2 *is derived from* s_1 *by a rewriting step in* Γ, *written as*

$$s_1 = (L_1, \ldots, L_n) \Rightarrow_\Gamma s_2 = (L'_1, \ldots, L'_n),$$

if $L_i = \{\alpha_{i_1}, \ldots, \alpha_{i_{r_i}}\}$ *and* $L'_i = \{\beta_{i_1}\} \cup \ldots \cup \{\beta_{i_{r_i}}\}$, *where* $\alpha_{i_k}, \beta_{i_k} \in V^*$, $\alpha_{i_k} \Rightarrow_{P_i} \beta_{i_k}$, $1 \leq i \leq n$, $1 \leq k \leq r_i$. *Furthermore,* $L'_i = \emptyset$ *in case* $L_i = \emptyset$, $1 \leq i \leq n$.

If no confusion arises, then Γ may be omitted from notation $\Rightarrow_\Gamma$.

Observe that the number of strings belonging to L'_i, $1 \leq i \leq n$, may be smaller than the number of strings of L_i, $1 \leq i \leq n$, owing to the fact that the same new string may be derived from two different strings.

Definition 4 *Let* $\Gamma = (V, (P_1, F_1, \varrho_1, \sigma_1), \ldots, (P_n, F_n, \varrho_n, \sigma_n))$, $n \geq 1$, *an* $\mathrm{NPLP}_{\mathrm{F0L}}$ *system,* $s_1 = (L_1, \ldots, L_n)$ *and* $s_2 = (L'_1, \ldots, L'_n)$ *two states (configurations) of* Γ, *where* $L_i, L'_i \subseteq V^*$, $1 \leq i \leq n$. s_2 *is derived from* s_1 *by a communication step in* Γ, *written as*

$$s_1 = (L_1, \ldots, L_n) \vdash_\Gamma s_2 = (L'_1, \ldots, L'_n),$$

if for every i, $1 \leq i \leq n$,

$$L'_i = L_i \cup \textstyle\bigcup_{j=1, j \neq i}^{n} A_{i,j},$$

where $A_{i,j} = \{v \mid v \in L_j, \varrho_j(v) = true \text{ and } \sigma_i(v) = true\}$.

Γ can be left out from notation $\vdash_\Gamma$, if it does not cause any ambiguity. A sequence of subsequent states determines a computation in Γ.

Definition 5 *Let* $\Gamma = (V, (P_1, F_1, \varrho_1, \sigma_1), \ldots, (P_n, F_n, \varrho_n, \sigma_n))$, $n \geq 1$, *an* $\mathrm{NPLP}_{\mathrm{F0L}}$ *system. By a computation* C *in* Γ, *we mean a sequence of states* $s_0, s_1, \ldots$, *where* $s_k \Rightarrow s_{k+1}$, *if* $k = 2j + 1, j \geq 0$, *and* $s_k \vdash s_{k+1}$, *if* $k = 2j, j \geq 1$.

A computation in Γ is finite provided that the sequence of states (configurations) is finite.

Languages can be associated with NPLP_{F0L} systems in various ways. One possibility is to collect all the strings obtained after a rewriting step into a language at a distinguished node, the master, in the course of a finite computation. Another potential in case the computation is finite, is to identify the language of an NPLP_{F0L} system with all the strings received at any nodes. Thus in the first case the language of Γ is as follows:

Definition 6 *Let* $\Gamma = (V, (P_1, F_1, \varrho_1, \sigma_1), \ldots, (P_n, F_n, \varrho_n, \sigma_n))$, $n \geq 1$, *an* NPLP_{F0L} *system. The language* $L(\Gamma)$ *of* Γ *is defined by* $L(\Gamma) = \{\omega \in L_1^{(s)} \mid (F_1, \ldots, F_n) = (L_1^{(0)}, \ldots, L_n^{(0)}) \Rightarrow (L_1^{(1)}, \ldots, L_n^{(1)}) \vdash (L_1^{(2)}, \ldots, L_n^{(2)}) \Rightarrow \ldots \Rightarrow (L_1^{(s)}, \ldots, L_n^{(s)}), s \geq 1\}$.

The language of Γ consists of all strings that are obtained at the master node after performing a rewriting step at the last step of a finite computation.

To illustrate the definitions let us consider the following example (see [34]):

Example 1 *Let* $L = \{a^{2^n}, a^{3^n} \mid n \geq 0\}$. *It is easy to see that* $L \notin \mathcal{L}(\text{T0L})$. *Language* L *can be generated the following* NPLP_{F0L} *system:*

$$\Gamma = (V, (P_1, F_1, \varrho_1, \sigma_1), (P_2, F_2, \varrho_2, \sigma_2), (P_3, F_3, \varrho_3, \sigma_3)),$$

where

- $V = \{a\}$;
- $P_1 = \{a \to a\}$, $P_2 = \{a \to a^2\}$, $P_3 = \{a \to a^3\}$;
- $F_1 = F_2 = F_3 = \{a\}$;
- $\varrho_1(u) = false$ *for* $u \in a^*$, *otherwise* $\varrho_1(u) = true$;
  $\sigma_1(u) = true$ *for* $u \in a^*$, *otherwise* $\sigma_1(u) = false$;
  $\varrho_2(u) = true$ *for* $u \in a^*$, *otherwise* $\varrho_2(u) = false$;
  $\sigma_2(u) = false$ *for* $u \in a^*$, *otherwise* $\sigma_2(u) = true$;

$\varrho_3(u) = true$ *for* $u \in a^*$, *otherwise* $\varrho_3(u) = false$;
$\sigma_3(u) = false$ *for* $u \in a^*$, *otherwise* $\sigma_3(u) = true$.

Component P_1 *collects strings of forms* a^{2^n} *and* a^{3^n}, $n \geq 0$, *but does not issue any string. Component* P_2 *produces strings of the form* a^{2^n}, $n \geq 0$, *and it does not accept any string. Component* P_3 *generates strings of the form* a^{3^n}, $n \geq 0$, *and it does not accept any string from the other components. As a consequence, it can be proven that* $L(\Gamma) = L$.

2.2.4 Simple Eco–Grammar Systems

In this section we overview the definition of simple eco–grammar systems. Due to the fact that in this dissertation we study the behaviour of some prototypes of self–organizing systems through simple eco–grammar systems, herein we do not present the general model. For the characterization of the general model, the reader is referred to [26, 31].

Definition 7 *A simple eco–grammar system (a* SEG *system) with* n *agents,* $n \geq 1$, *is a construct*

$$\Gamma = (V_E, P_E, R_1, \ldots, R_n, \omega),$$

where

- V_E *is a finite alphabet, the alphabet of the system,*
- P_E *is a finite and complete set of pure context–free rules over* V_E *(i.e. rules of the form* $a \rightarrow \alpha$ *with* $a \in V_E$, $\alpha \in V_E^*$, *and for each* $a \in V_E$, *there is a rule* $a \rightarrow \alpha$ *in* P_E*), the set of developmental rules of the environment,*
- $R_i, 1 \leq i \leq n$, *is a finite set of pure context–free rules over* V_E, *the set of action rules of the* i*–th agent,*
- $\omega \in V_E^+$ *is the axiom, the initial state of the environment.*

A string over V_E^* is called the state of the environment or the environmental state. A SEG system functions through the change of the environmental state. A SEG system works in such a manner that the agents substitute exactly one occurrence of a symbol with a word by means of their action rules in the current environmental state, whereas the other symbols are rewritten by the developmental rules of the environment in a parallel way.

Definition 8 *Let* $\Gamma = (V_E, P_E, R_1, \ldots, R_n, \omega)$ *be a* SEG *system.* ω_E *directly derives* ω'_E *in* Γ, $\omega_E, \omega'_E \in V_E^*$, *written as* $\omega_E \Rightarrow_\Gamma \omega'_E$, *if*

1. $\omega_E = x_1 a_1 x_2 \ldots x_m a_m x_{m+1}$ *and* $\omega'_E = y_1 z_1 y_2 \ldots y_m z_m y_{m+1}$, *for some* m, $0 \leq m \leq n$, $a_h \in V_E, x_j, y_j, z_h \in V_E^*$, $1 \leq h \leq m, 1 \leq j \leq m+1$,
2. $a_h \rightarrow z_h \in R_{i_h}$, $\{i_1, \ldots, i_h\} \subseteq \{1, \ldots, n\}$, $1 \leq h \leq m$,
3. $y_j = x_j$ *is either the empty word, or* $x_j \Rightarrow_{P_E} y_j, 1 \leq j \leq m+1$, *is a* 0L *rewriting.*

The transitive, reflexive closure of $\Rightarrow_\Gamma$ is denoted by $\overset{*}{\Rightarrow}_\Gamma$.

The language of a SEG system is the set of all environmental states that are reachable from the initial state.

Definition 9 *Let* $\Gamma = (V_E, P_E, R_1, \ldots, R_n, \omega)$ *be a* SEG *system. The generated language is defined by* $L(\Gamma) = \{y \mid \omega \overset{*}{\Rightarrow}_\Gamma y\}$.

To illustrate how SEG systems work, we consider the following example:

Example 2 *Let the* SEG *system be*

$$\Gamma = (V_E, P_E, R_1, R_2, R_3, \omega),$$

where $V_E = \{a, b, c\}$, $P_E = \{a \rightarrow a, b \rightarrow b, c \rightarrow c\}$, $R_1 = \{a \rightarrow a^2\}$, $R_2 = \{b \rightarrow b^2\}$, $R_3 = \{c \rightarrow c^2\}$, $\omega = abc$. *It is easy to see that the generated language is* $L(\Gamma) = \{a^n b^n c^n\}$, *which is not context–free.*

The extended version of SEG systems, where a subalphabet T_E of V_E is distinguished, was introduced in [31]:

Definition 10 *An extended* SEG *system with n agents, $n \geq 1$, is a construct*

$$\Gamma = (V_E, T_E, P_E, R_1, \ldots, R_n, \omega),$$

where

- *V_E, P_E, $R_i, 1 \leq i \leq n$, ω, are the same as in Def. 7, i.e. the alphabet of the system, the set of developmental rules of the environment, the set of action rules of the i–th agent, the initial state of the environment,*
- *$T_E \subseteq V_E$ is the terminal alphabet of the system.*

In an extended SEG system, the derivation occurs in an analogous manner to how it is given in Def. 8. The generated language, however, contains only words obtainable from the axiom over the terminal alphabet.

Chapter 3

Peer–to–Peer Networks

In this chapter we propose a variant of networks of parallel language processors (see, e.g. [27, 28, 34, 110]) to describe the behaviour of peer–to–peer (P2P) systems. In our model the members of the P2P networks are represented by multiset string processors, form teams, send and receive information through collective and individual filters. Our work deals with the dynamics of string collections. The connection between the growth of the number of strings being present during the computation at the components of the network and the growth function of a developmental system is also established. We demonstrate how the formal language theoretic model can be employed to incorporate network security requirements. More specifically, we show how to model and detect SYN flooding attacks [105] and enforce Discretionary Access Control [113]. The results of this chapter appeared in [73] and [77].

The organization of this chapter is as follows. In Section 3.1, we overview P2P networking in general. In Section 3.2, we present the formal language theoretic construction to model P2P networks and focus on the dynamics of string collections in these systems. In Section 3.3, we illustrate how to detect SYN flooding attacks and enforce Discretionary Access Control. Finally, in Section 3.4, we conclude this chapter and suggest some future work.

3.1 Introduction

Nowadays the Internet is witnessing a revolution hailed as peer–to–peer networking. P2P networks are defined differently in the literature (for further details about the current P2P principles consult [1] and references therein). In our work, we rely on the definitions and standards of the JXTA–based P2P systems [46, 121].

The P2P architecture enables true distributed computing, creating networks of computing resources that can exhibit very high availability and fault tolerance. Peer–to–peer networking promises to create a computing world substantially different from the one based on the traditional client/server model. In P2P systems, the entities, referred to as *peers*, have equal status, meaning that a peer can either request a service (a client trait) or provide a service (a server trait). Peers can include sensors, phones, and PDAs, as well as PCs, servers and supercomputers. Each peer, uniquely identified by a Peer ID, operates independently, communicates with other peers asynchronously and can have its own non–peer clients.

A *peer group* is a collection of peers that have agreed upon a common set of services. According to recent developments (see, e.g. [121]), peers can self–organize themselves into peer groups, identified by a unique peer group ID, so as to create a secure, a scoping and a monitoring environment. Each peer group can establish its own membership policy ranging from open (anybody can join) to highly secure and protected (sufficient credentials are required to join). Peers may belong to more than one peer group simultaneously.

Both peers and peer groups can offer network services. A *peer service* is accessible only on the peer that publishes the service. Should that peer fail, the service also fails. A *peer group service* is composed of a collection of instances (cooperating with each other) of the service running on multiple members of the peer group. If a peer fails, the collective peer group service is not affected (assuming the service is still available from another peer member). Peer group services are published as part of the peer group advertisement.

A *message* is the basic unit of data exchange between peers. Peers utilize *pipes* to send messages to each other. A pipe is an asynchronous and unidirectional message transfer

mechanism used for service communication. The pipe endpoints are referred to as the input pipe (the receiving end) and the output pipe (the sending end). A *point–to–point pipe* connects exactly two pipe endpoints together: an input pipe on one peer receives messages sent from the output pipe of another peer. There are several possibilities, such as the *propagate pipe*, which connects one output pipe to multiple input pipes within the same peer group, or the *secure unicast pipe*, a point–to–point pipe that provides a secure communication channel. Bidirectional pipes and bidirectional/reliable pipes can also be constructed.

All network resources – such as peers, peer groups, pipes, and services – are represented by *advertisements* used to describe and publish the existence of a peer resource. Peers discover resources by searching for their corresponding advertisements and may cache any discovered advertisements locally. Advertisements may have very short lifetimes: they are published with time limits about the availability of their associated resources. Owing to this method and similar ones, centralized control will not be needed any more. The self–organization of the peer groups is supposed to take care of spatiotemporal information propagation. For example, an advertisement can be republished (before the original advertisement expires) to extend the lifetime of a resource. The central concept is the *rendezvous advertisement*, which describes a peer acting as a rendezvous peer for a given peer group: it is a special peer that stores information about other peers by caching advertisements of the known peers and as a result, it can help them discover other peers in the network.

In this chapter we are going to study the peer–to–peer communication in a formal language theoretic framework, called networks of parallel multiset string processors with teams of collective and individual filtering. In our model a peer is represented by a multiset string processor, situated at a given node of the network and employs various filters for information transmission. Like peers in a P2P system, multiset string processors possess identical functionalities. They form teams, which correspond to peer groups. Both multiset string processors and teams have unique IDs. The components of the network operate on multisets of strings corresponding to advertisements or messages, by performing rewriting or communication steps alternately.

The formal language theoretic construction introduced herein, is a simplified abstract model

of a P2P network, but it contains the most significant properties of such systems. Clearly, more sophisticated features could be added, which might cast a new light on some additional aspects of P2P networking. The aim of this chapter is twofold. First, it gives a formal language theoretic model of P2P systems. Secondly, it characterizes the dynamics of information in the network and discusses some related state–of–the–art issues.

3.2 Formal Definitions

In the sequel, we introduce the notion of network of parallel multiset string processors with teams of collective and individual filtering and define the way in which such a system works.

Definition 11 *A network of parallel multiset string processors with teams of collective and individual filtering (a* $\mathrm{T_{ci}NPMP_{F0L}}$ *system) of degree* n, $n \geq 1$, *is a construct*

$$\Gamma = (V, (t_1, \Theta_1, \Xi_1), \ldots, (t_n, \Theta_n, \Xi_n)),$$

where

- V *is an alphabet, the alphabet of the system,*
- $t_i = \{c_{i,1}, \ldots, c_{i,r_i}\}, 1 \leq i \leq n, r_i \geq 1$, *is a team component, the* i*–th team, where*
  - $c_{i,j} = (P_{i,j}, F_{i,j}, \Psi_{i,j}, \Upsilon_{i,j}), 1 \leq i \leq n, 1 \leq j \leq r_i$, *is the* j*–th component of the* i*–th team of the network, or in other words, the* (i,j)*–th component of the network, where*
    - $P_{i,j}$, $1 \leq i \leq n, 1 \leq j \leq r_i$, *is a finite and complete set of pure context–free rules over* V *(i.e. rules of the form* $A \to \alpha$ *with* $A \in V$, $\alpha \in V^*$, *and for each* $A \in V$, *there is a rule* $A \to \alpha$ *in* $P_{i,j}$*), the production set of the* (i,j)*–th component,*
    - $F_{i,j} \in V^\circ$, $1 \leq i \leq n$, $1 \leq j \leq r_i$, *is a non–empty finite multiset of strings, the multiset of axioms of the* (i,j)*–th component, and*

* $\Psi_{i,j} = \{\psi_{i,j_1}, \ldots, \psi_{i,j_{s_{i,j}}}\}$, $\Upsilon_{i,j} = \{\upsilon_{i,j_1}, \ldots, \upsilon_{i,j_{o_{i,j}}}\}$, $1 \leq i \leq n, 1 \leq j \leq r_i$, *where* $\psi_{i,j_k}, \upsilon_{i,j_l}$, $1 \leq k \leq s_{i,j}, 1 \leq l \leq o_{i,j}$, *are context conditions over* V^*, *called an exit filter and an entrance filter, respectively, of the* (i,j)*–th component,*

- $\Theta_i = \{\theta_{i1}, \ldots, \theta_{ip_i}\}$, $\Xi_i = \{\xi_{i1}, \ldots, \xi_{iq_i}\}$, $1 \leq i \leq n$, *where* θ_{ij}, ξ_{ik}, $1 \leq j \leq p_i, 1 \leq k \leq q_i$, *are context conditions over* V^*, *called an exit filter and an entrance filter, respectively, of the* i*–th team.*

A component or a multiset string processor corresponds to a peer, while a team to a peer group in a P2P system.

An element of $F_{i,j} \in V^\circ$, $1 \leq i \leq n, 1 \leq j \leq r_i$, may either correspond to an advertisement or a message. Whether the underlying string is an advertisement or a message, might be expressed by a disjoint alphabet, but it has no impact on the mathematical results established in this chapter, thus it is omitted. Using the terminology of networks of language processors, the component, in effect, is a multiset string processor. The choice of a multiset string processor is motivated by the fact that in P2P networks multiple instances of an advertisement or a message may exist on the members of a peer group, and each receiver of an advertisement or a message takes away its own copy.

In the case of an advertisement, filters θ_{ij}, ξ_{ik}, $1 \leq i \leq n, 1 \leq j \leq p_i, 1 \leq k \leq q_i$, limit access to advertisements available to every multiset string processor (collective filtering of information), whilst filters $\psi_{i,j_k}, \upsilon_{i,j_l}$, $1 \leq i \leq n, 1 \leq j \leq r_i, 1 \leq k \leq s_{i,j}, 1 \leq l \leq o_{i,j}$, to those advertisements that are available only to the components of the given team (individual filtering of information).

In the case of a message, filters θ_{ij}, ξ_{ik}, $1 \leq i \leq n, 1 \leq j \leq p_i, 1 \leq k \leq q_i$, are the pipe endpoints referred to as the output pipe (the sending end) and as the input pipe (the receiving end) at collective information filtering level, whereas filters $\psi_{i,j_k}, \upsilon_{i,j_l}, 1 \leq i \leq n, 1 \leq j \leq r_i, 1 \leq k \leq s_{i,j}, 1 \leq l \leq o_{i,j}$, are the pipe endpoints referred to as the output pipe (the sending end) and as the input pipe (the receiving end) at individual information filtering level, respectively.

According to the type of the filters and the type of the productions sets we distinguish different classes of $\mathrm{T_{ci}NPMP}$ systems. We denote by $\mathrm{T_{c_X i_Y}NPMP_Z}$ the class of $\mathrm{T_{ci}NPMP}$ systems with (X)–type collective and (Y)–type individual filters, where $X, Y \in \{reg, rc\}$ and $Z \in \{$0L, D0L, F0L, ... $\}$.

The $\mathrm{T_{ci}NPMP_{F0L}}$ system functions by changing its states.

Definition 12 *By a state (or a configuration) of a* $\mathrm{T_{ci}NPMP_{F0L}}$ *system* $\Gamma = (V, (t_1, \Theta_1, \Xi_1), \ldots, (t_n, \Theta_n, \Xi_n))$, $n \geq 1$, *as above (see Def. 11), we mean a tuple* $s = (M_{1,1}, \ldots, M_{1,r_1}, \ldots, M_{n,1}, \ldots, M_{n,r_n})$, *where* $M_{i,j} \in V^\circ$, $1 \leq i \leq n, 1 \leq j \leq r_i$, *is called the state of the* (i,j)*–th component and it represents the multiset of strings present at component* (i,j) *at that step.* $s_0 = (F_{1,1}, \ldots, F_{1,r_1}, \ldots, F_{n,1}, \ldots, F_{n,r_n})$ *is called the initial state of the system.*

Definition 13 *(Configuration transmission.)*
Let $\Gamma = (V, (t_1, \Theta_1, \Xi_1), \ldots, (t_n, \Theta_n, \Xi_n))$, $n \geq 1$, *be a* $\mathrm{T_{ci}NPMP_{F0L}}$ *system, as above (see Def. 11). Let* $s_1 = (M_{1,1}, \ldots, M_{1,r_1}, \ldots, M_{n,1}, \ldots, M_{n,r_n})$ *and* $s_2 = (M'_{1,1}, \ldots, M'_{1,r_1}, \ldots, M'_{n,1}, \ldots, M'_{n,r_n})$ *be two states of* Γ. *We say that*

1. s_2 *is derived from* s_1 *by a rewriting step in* Γ, *written as*

$$(M_{1,1}, \ldots, M_{1,r_1}, \ldots, M_{n,1}, \ldots, M_{n,r_n}) \Rightarrow (M'_{1,1}, \ldots, M'_{1,r_1}, \ldots, M'_{n,1}, \ldots, M'_{n,r_n}),$$

   if $M_{i,j} = \{\{\alpha_{i,j_1}, \ldots, \alpha_{i,j_{g_{i,j}}}\}\}, M'_{i,j} = \{\{\beta_{i,j_1}, \ldots, \beta_{i,j_{g_{i,j}}}\}\}$, *where* $\alpha_{i,j_k}, \beta_{i,j_k} \in V^*$, $\alpha_{i,j_k} \Rightarrow \beta_{i,j_k}$ *in* $P_{i,j}$, $1 \leq i \leq n, 1 \leq j \leq r_i$, $1 \leq k \leq g_{i,j}$.

2. s_2 *is derived from* s_1 *by a communication step in* Γ, *written as*

$$(M_{1,1}, \ldots, M_{1,r_1}, \ldots, M_{n,1}, \ldots, M_{n,r_n}) \vdash (M'_{1,1}, \ldots, M'_{1,r_1}, \ldots, M'_{n,1}, \ldots, M'_{n,r_n}),$$

   if for every $1 \leq i \leq n$, $1 \leq j \leq r_i$,

$$M'_{i,j} = M_{i,j} \cup C_{i,j} \cup I_{i,j},$$

   where

$$C_{i,j} = \{\{\gamma \mid \gamma \in M_{k,l}, \theta_{kx}(\gamma) = true, \xi_{iy}(\gamma) = true, 1 \leq k \leq n, 1 \leq l \leq r_k,$$
$$1 \leq x \leq p_k, 1 \leq y \leq q_i, (k,l) \neq (i,j)\}\}, \textit{ and}$$
$$I_{i,j} = \{\{\gamma \mid \gamma \in M_{i,k}, \psi_{i,k_u}(\gamma) = true, \upsilon_{i,j_v}(\gamma) = true, 1 \leq k \leq r_i,$$
$$1 \leq u \leq s_{i,k}, 1 \leq v \leq o_{i,j}, j \neq k\}\}.$$

If the underlying string is an advertisement, then Cond. 1 of Def. 13 corresponds to the publication or the update of the advertisement. Should the string be a message, then Cond. 1 refers to the compilation or the modification of the message. We apply parallel rewriting rules, since the entire advertisement or message can be modified at a given time step. As a result of the rewriting step, only one new string can be derived from each string through the application of 0L productions. Moreover, some of these strings may be identical.

Cond. 2 of Def. 13, if $C_{i,j} \neq \epsilon$ and $I_{i,j} = \epsilon$, then component $c_{i,j}$ performs the collective, if $C_{i,j} = \epsilon$ and $I_{i,j} \neq \epsilon$, then the individual, if $C_{i,j} \neq \epsilon$ and $I_{i,j} \neq \epsilon$, then the simultaneous collective and individual filtering mechanism. If $I_{i,j} = C_{i,j} = \epsilon$, then none of the strings is allowed to penetrate the entrance filters of $c_{i,j}$ and the entrance filters of the team $c_{i,j}$ belongs to.

The components communicate the copies of the strings at their disposal. If the string to be communicated is an advertisement, then a component can apply either for an advertisement that may be available to arbitrary member of an arbitrary team (collective filtering mechanism), for an advertisement that may be available only to the members of the team the given component belongs to (individual filtering mechanism), or for both of the previous two types of advertisements (simultaneous collective and individual filtering mechanism), in case some context conditions are met. Should the string to be communicated be a message, it might been transferred either via the pipe that connects two components belonging to arbitrary teams (collective filtering mechanism), via the pipe that connects two members of the team the given component belongs to (individual filtering mechanism), or via both of the previous two types of pipes (simultaneous collective and individual filtering mechanism), provided that some context conditions are satisfied. In the case of a message, the satisfiability of the given context condition means that the component intent on sending/receiving the message is able to send/receive it.

A sequence of subsequent states determines a computation in Γ.

Let $\Gamma = (V, (t_1, \Theta_1, \Xi_1), \ldots, (t_n, \Theta_n, \Xi_n))$, $n \geq 1$, be a $\mathrm{T_{ci}NPMP_{F0L}}$ system. By a computation C in Γ we mean a sequence of states $s_0, s_1, \ldots$, where $s_k \Rightarrow s_{k+1}$, if $k = 2j+1, j \geq 0$, and $s_k \vdash s_{k+1}$, if $k = 2j, j \geq 1$.

3.2.1 Information Dynamics

By using the previous formalism, in the following we show how the dynamics of information can be characterized in P2P networks.

Definition 14 *Let $\Gamma = (V, (t_1, \Theta_1, \Xi_1), \ldots, (t_n, \Theta_n, \Xi_n))$, $n \geq 1$, be a $\mathrm{T_{c_{rc}i_{rc}}NPMP_{FD0L}}$ system and let $(M_{1,1}^{(t)}, \ldots, M_{1,r_1}^{(t)}, \ldots, M_{n,1}^{(t)}, \ldots, M_{n,r_n}^{(t)})$ be the state of Γ at step t during the computation in Γ, where $t \geq 0, r_i \geq 1, 1 \leq i \leq n$.*

1. *Function $m : \mathbb{N}_0 \to \mathbb{N}_0$ defined by $m(t) = \sum_{i=1}^{n} \sum_{j=1}^{r_i} card(M_{i,j}^{(t)})$, for $t \geq 0$, is called the population growth function of Γ.*

2. *Function $m_{i,j} : \mathbb{N}_0 \to \mathbb{N}_0$ defined by $m_{i,j}(t) = card(M_{i,j}^{(t)})$, for $t \geq 0$, is called the population growth function of Γ at node (i,j), $1 \leq i \leq n, 1 \leq j \leq r_i$.*

3. *(Communication functions.)*

   (a) *Function $f_{(i,j)(k,l)}^{c} : \mathbb{N}_0 \to \mathbb{N}_0$ defined by $f_{(i,j)(k,l)}^{c}(t) = card(\{\{\gamma \in M_{i,j}^{(t-1)} \mid \theta_{ix}(\gamma) = true, \xi_{ky}(\gamma) = true, 1 \leq k \leq n, 1 \leq l \leq r_k, 1 \leq x \leq p_i, 1 \leq y \leq q_k, (k,l) \neq (i,j)\}\})$, for $t = 2k', k' \geq 1$, and $f_{(i,j)(k,l)}^{c}(t) = 0$ otherwise, is called the communication function of Γ from node (i,j) to node (k,l) using collective filtering.*

   (b) *Function $f_{(i,j)(i,k)}^{i} : \mathbb{N}_0 \to \mathbb{N}_0$ defined by $f_{(i,j)(i,k)}^{i}(t) = card(\{\{\gamma \in M_{i,j}^{(t-1)} \mid \psi_{i,j_u}(\gamma) = true, \upsilon_{i,k_v}(\gamma) = true, 1 \leq k \leq r_i, 1 \leq u \leq s_{i,j}, 1 \leq v \leq o_{i,k}, j \neq k\}\})$, for $t = 2k', k' \geq 1$, and $f_{(i,j)(i,k)}^{i}(t) = 0$ otherwise, is called the communication function of Γ from node (i,j) to node (i,k) using individual filtering.*

Since in Def. 14 Γ is a $\mathrm{T}_{\mathrm{c_{rc}i_{rc}}}\mathrm{NPMP}_{\mathrm{FD0L}}$ system, only one string may be derived from each string during the rewriting step. Therefore, m and m_i can be regarded as functions. The population growth function of Γ, m, describes the increase in the number of pieces of information in the network, the population growth function of Γ at node (i,j), $m_{i,j}$, the increase in the number of pieces of information at node (i,j), and the communication function of Γ from node (i,j) to node (k,l) $((i,k))$, $f^c_{(i,j)(k,l)}$ $(f^i_{(i,j)(i,k)})$, the increase in the number of pieces of information during the communication between node (i,j) and node (k,l) $((i,k))$ using collective (individual) filtering at a given time step, respectively.

We demonstrate that the change of the rewritten and the communicated string collections using random context filters can be described by developmental systems.

Theorem 1 *Let $\Gamma = (V, (t_1, \Theta_1, \Xi_1), \ldots, (t_n, \Theta_n, \Xi_n))$, $n \geq 1$, be a $\mathrm{T}_{\mathrm{c_{rc}i_{rc}}}\mathrm{NPMP}_{\mathrm{FD0L}}$ system. Then a* D0L *system $H = (\Sigma, \omega, h)$ can be constructed, such that*

1. *$m(t) = f(t)$, where m is the population growth function of Γ and f is the growth function of H;*
2. *$m_{i,j}(t) = card(\bar{h}_{i,j}(h^t(\omega)))$ for some erasing homomorphism $\bar{h}_{i,j} : \Sigma \to \Sigma$, where $m_{i,j}$ is the population growth function of Γ at node (i,j);*
3. *(communication functions)*
   (a) *$f^c_{(i,j)(k,l)}(t) = card(\bar{h}_{(i,j)(k,l)}(h^t(\omega)))$ for some erasing homomorphism $\bar{h}_{(i,j)(k,l)} : \Sigma \to \Sigma$, where $f^c_{(i,j)(k,l)}$ is the communication function of Γ from node (i,j) to node (k,l), $t \geq 0, 1 \leq i,k \leq n, 1 \leq j \leq r_i, 1 \leq l \leq r_k, (k,l) \neq (i,j)$, using collective filtering;*
   (b) *$f^i_{(i,j)(i,k)}(t) = card(\bar{h}_{(i,j)(i,k)}(h^t(\omega)))$ for some erasing homomorphism $\bar{h}_{(i,j)(i,k)} : \Sigma \to \Sigma$, where $f^i_{(i,j)(i,k)}$ is the communication function of Γ from node (i,j) to node (i,k), $t \geq 0, 1 \leq i \leq n, 1 \leq j,k \leq r_i, j \neq k$, using individual filtering.*

Proof

1. Due to the fact that D0L systems define homomorphisms and the number of strings with a fixed minimal alphabet at a node is known, the number of strings with the same minimal alphabet at this node can be calculated after performing a rewriting

step. Context conditions check the presence and/or the absence of some symbols in the string. Since the (minimal) alphabet of the string is known, it is decidable whether the given string satisfies the underlying context condition, and consequently, any multiset of string present at some stage of computation in Γ can be represented by a multiset of symbols identifying the different alphabets in a unique manner.

It can be shown that at any computation step (both at the rewriting and at the communication step) in Γ, the multiset of these symbols equals the multiset of letters of a word of a D0L system H, which generates only the words that represent the states (string collections) of Γ in the above described manner.

To prove the statement, we construct a D0L system $H = (\Sigma, \omega, h)$.

Let homomorphisms $h_{i,1}, \ldots, h_{i,r_i}$ be defined by production sets $P_{i,1}, \ldots, P_{i,r_i}$, $1 \leq i \leq n, r_i \geq 1$, of Γ and let context conditions θ_{ix}, ξ_{iy}, and $\psi_{i,j_u}, \upsilon_{i,j_v}$, $1 \leq i \leq n, 1 \leq j \leq r_i, 1 \leq x \leq p_i, 1 \leq y \leq q_i, 1 \leq u \leq s_{i,j}, 1 \leq v \leq o_{i,j}$, be expressed explicitly by notations (Q_{ix}, R_{ix}), (Q_{iy}, R_{iy}) and (Q_{i,j_u}, R_{i,j_u}), (Q_{i,j_v}, R_{i,j_v}), respectively, where $Q_{ix}, Q_{iy}, Q_{i,j_u}, Q_{i,j_v}$ are the corresponding sets of permitting and $R_{ix}, R_{iy}, R_{i,j_u}, R_{i,j_v}$, $1 \leq i \leq n, 1 \leq j \leq r_i, 1 \leq x \leq p_i, 1 \leq y \leq q_i, 1 \leq u \leq s_{i,j}, 1 \leq v \leq o_{i,j}$, are the corresponding sets of forbidding symbols, respectively.

Let $\{V_1, \ldots, V_{2^m}\}$ be the set of subsets of V, where $m = card(V)$, and let $\Sigma = \{a_{i,j_w}, b_{i,j_w} \mid 1 \leq i \leq n, 1 \leq j \leq r_i, 1 \leq w \leq 2^m\}$.

For the sake of legibility, instead of defining homomorphism h of H, the corresponding production set P is presented.

For every $i, j, w, 1 \leq i \leq n, 1 \leq j \leq r_i, 1 \leq w \leq 2^m$, the construction of the rules of P is as follows:

(a) $a_{i,j_w} \rightarrow b_{i,j_z}$, if $alph(h_{i,j}(V_w)) = V_z$, where $1 \leq z \leq 2^m$.

(b) $b_{i,j_w} \rightarrow a_{i,j_w} a_{d_1,k_{1_w}} \ldots a_{d_{s'},k_{s'_w}} a_{i,e_{1_w}} \ldots a_{i,e_{s_w}}$, $1 \leq d_l \leq n, 1 \leq k_l \leq r_{d_l}, 1 \leq l \leq s'$, $1 \leq e_{l''} \leq r_i, 1 \leq l'' \leq s$, where

   i. $(d_l, k_l) \neq (d_{l'}, k_{l'})$, if $l \neq l'$ (pairwise different), and $\{(d_1, k_1), \ldots, (d_{s'}, k_{s'})\}$ is the maximal subset of $\{(1,1) \ldots, (1, r_1), \ldots, (n, 1) \ldots, (n, r_n)\} - \{(i, j)\}$, such that for every (d_l, k_l) there are x and y, $1 \leq x \leq p_i, 1 \leq y \leq q_{d_l}, 1 \leq d_l \leq n, 1 \leq k_l \leq r_{d_l}, 1 \leq l \leq$

s', for which it holds that $Q_{ix} \subseteq V_w, R_{ix} \cap V_w = \emptyset$ and $Q_{d_l y} \subseteq V_w, R_{d_l y} \cap V_w = \emptyset$, i.e. components $c_{i,j}$ and c_{d_l,k_l} use the collective filtering mechanism, and

ii. $e_1, \ldots, e_s, s \geq 1$, are pairwise different numbers, and $\{e_1, \ldots, e_s\}$ is the maximal subset of $\{1, \ldots, r_i\} - \{j\}$, such that for every $e_{l''}$ there are u and v, $1 \leq u \leq s_{i,j}, 1 \leq v \leq o_{i,e_{l''}}, 1 \leq e_{l''} \leq r_i, 1 \leq l'' \leq s$, for which it holds that $Q_{i,j_u} \subseteq V_w, R_{i,j_u} \cap V_w = \emptyset$ and $Q_{i,e_{l''}v} \subseteq V_w, R_{i,e_{l''}v} \cap V_w = \emptyset$, i.e. components $c_{i,j}$ and $c_{i,e_{l''}}$ use the individual filtering mechanism.

Should there be no (d_l, k_l), no $e_1, \ldots, e_s$, or no (d_l, k_l) and $e_1, \ldots, e_s$, $1 \leq d_l \leq n, 1 \leq k_l \leq r_{d_l}, 1 \leq l \leq s', s \geq 1$, with the above properties, then P has production of the form $b_{i,j_w} \rightarrow a_{i,j_w} a_{i,e_{1_w}} \ldots a_{i,e_{s_w}}$, $b_{i,j_w} \rightarrow a_{i,j_w} a_{d_1,k_{1_w}} \ldots a_{d_{s'},k_{s'_w}}$, or $b_{i,j_w} \rightarrow a_{i,j_w}$, respectively, where $1 \leq d_l \leq n, 1 \leq k_l \leq r_{d_l}, 1 \leq l \leq s'$, $1 \leq e_{l''} \leq r_i, 1 \leq l'' \leq s$.

Let $F_{i,j} = \{\{v_{i,j_1}, \ldots, v_{i,j_{m_{i,j}}}\}\}$, where $m_{i,j} = card(F_{i,j}), 1 \leq i \leq n, 1 \leq j \leq r_i$. Let

$$g(F_{i,j}) = \begin{cases} g(v_{i,j_1}) \ldots g(v_{i,j_{m_{i,j}}}), & \text{if } F_{i,j} \neq \emptyset, \\ \lambda, & \text{otherwise,} \end{cases}$$

where $g(v_{i,j_w}) = a_{i,j_z}$, if $alph(v_{i,j_w}) = V_z, 1 \leq z \leq 2^m$.

Let $\omega = g(F_{1,1}) \ldots g(F_{1,r_1}) \ldots g(F_{n,1}) \ldots g(F_{n,r_n})$.

We show that the growth function of H equals the population growth function of Γ. Obviously, symbol $a_{i,j_k}, 1 \leq i \leq n, 1 \leq j \leq r_i$, in ω corresponds to a word in $F_{i,j}$ with alphabet V_k, and reversely. Consequently, the length of ω equals the number of axioms of Γ.

In fact, productions given by Cond. 1a describe a rewriting step in Γ: a word of alphabet V_z is derived by means of $P_{i,j}$ from a word of alphabet V_w. The productions are applied in a parallel manner, thus all strings are represented after a rewriting step at node (i,j), $1 \leq i \leq n, 1 \leq j \leq r_i$.

Productions given by Cond. 1b, on the other hand, describe the communication step. If a word of alphabet V_w at node (i,j), $1 \leq i \leq n, 1 \leq j \leq r_i$, can be communicated to nodes $(d_1, k_1), \ldots, (d_{s'}, k_{s'})$ (i.e. components $c_{i,j}$ and c_{d_l,k_l} use the collective filtering mechanism), and/or to nodes $(i, e_1), \ldots, (i, e_s)$ (i.e. components $c_{i,j}$ and $c_{i,e_{l''}}$ use

the individual filtering mechanism), where $1 \leq d_l \leq n, 1 \leq k_l \leq r_{d_l}, 1 \leq l \leq s'$, $1 \leq e_{l''} \leq r_i, 1 \leq l'' \leq s$, then a new word, or in other words, a copy of the string over V_w will appear at those nodes, otherwise the underlying word will remain at the given node.

Provided that ω_t is the t–th member of the D0L sequence of H, then the length of ω_t equals the total number of strings present at the nodes at step t during a computation in Γ. Thus $m(t) = f(t)$ holds.

2. By choosing $\bar{h}_{i,j} : \Sigma \to \Sigma$ as $\bar{h}_{i,j}(a_{i,j_w}) = a_{i,j_w}, \bar{h}_{i,j}(b_{i,j_w}) = b_{i,j_w}$, $1 \leq i \leq n, 1 \leq j \leq r_i, 1 \leq w \leq 2^m$ and $\bar{h}_{i,j}(a_{k,l_w}) = \lambda$, $\bar{h}_{i,j}(b_{k,l_w}) = \lambda$, $1 \leq k \leq n, 1 \leq l \leq r_k, (k,l) \neq (i,j), 1 \leq w \leq 2^m$, the result is immediately obtained.

3. By choosing

   (a) $\bar{h}_{(i,j)(k,l)} : \Sigma \to \Sigma$ as $\bar{h}_{(i,j)(k,l)}(a_{k',l'_w}) = \lambda, \bar{h}_{(i,j)(k,l)}(b_{k'',l''_z}) = \lambda, \bar{h}_{(i,j)(k,l)}(b_{i,j_z}) = a_{k,l_z}$, if $b_{i,j_z} \to a_{i,j_z}\alpha a_{k,l_z}\beta, \alpha, \beta \in \Sigma^*$, and λ otherwise, $1 \leq i, k, k', k'' \leq n, 1 \leq j \leq r_i, 1 \leq l \leq r_k, 1 \leq l' \leq r_{k'}, 1 \leq l'' \leq r_{k''}, (i,j) \neq (k,l), (i,j) \neq (k'',l''), 1 \leq w, z \leq 2^m$, and

   (b) $\bar{h}_{(i,j)(i,k)} : \Sigma \to \Sigma$ as $\bar{h}_{(i,j)(i,k)}(a_{k',l'_w}) = \lambda, \bar{h}_{(i,j)(i,k)}(b_{k'',l''_z}) = \lambda, \bar{h}_{(i,j)(i,k)}(b_{i,j_z}) = a_{i,k_z}$, if $b_{i,j_z} \to a_{i,j_z}\alpha a_{i,k_z}\beta, \alpha, \beta \in \Sigma^*$, and λ otherwise, $1 \leq i, k', k'' \leq n, 1 \leq j, k \leq r_i, 1 \leq l' \leq r_{k'}, 1 \leq l'' \leq r_{k''}, k \neq j, (i,j) \neq (k'',l''), 1 \leq w, z \leq 2^m$,

   the statements follow. ∎

Theorem 1 describes how to construct a communication graph by means of communication functions, since the sequence of communication functions with respect to a given time step defines a sequence of communication graphs.

By the theory of D0L systems (see, [109], for details), we obtain the following corollaries:

Corollary 1 *Let* $\Gamma = (V, (t_1, \Theta_1, \Xi_1), \ldots, (t_n, \Theta_n, \Xi_n))$, $n \geq 1$, *be a* $\mathrm{T}_{\mathrm{c_{rc}i_{rc}}}\mathrm{NPMP}_{\mathrm{FD0L}}$ *system. Then the population growth function of* Γ *is either exponential or polynomially bounded, which is decidable.*

Corollary 2 *Let* $\Gamma = (V, (t_1, \Theta_1, \Xi_1), \ldots, (t_n, \Theta_n, \Xi_n))$, $n \geq 1$, *be a* $\mathrm{T}_{\mathrm{crc_{irc}}}\mathrm{NPMP}_{\mathrm{FD0L}}$ *system. Suppose that* $H = (\Sigma, \omega, h)$ *is the* D0L *system for which conditions 1, 2 and 3 of Theorem 1 hold. Let* $\omega = \omega_0, \omega_1, \omega_2, \ldots$, *be the word sequence generated by the* D0L *system* H. *Then the sets* $\Sigma_i = alph(\omega_i), i \geq 0$, *form an almost periodic sequence, i.e. there are numbers* $p > 0$ *and* $q \geq 0$, *such that* $\Sigma_i = \Sigma_{i+p}$ *holds for every* $i \geq q$. *If a letter* $a \in \Sigma$ *occurs in some* Σ_i, *then it also appears in some* Σ_j, *with* $j \leq card(\Sigma) - 1$.

According to Corollary 2, we can claim that after some time the function of these P2P networks results in the saturation of information.

Corollary 3 *Let* $\Gamma_z = (V_z, (t_{1_z}, \Theta_{1_z}, \Xi_{1_z}), \ldots, (t_{n_z}, \Theta_{n_z}, \Xi_{n_z}))$, $n \geq 1$, *be a* $\mathrm{T}_{\mathrm{crc_{irc}}}\mathrm{NPMP}_{\mathrm{FD0L}}$ *system for* $z = 1, 2$. *Then the sequence and language equivalence problems are decidable for the* D0L *systems* $H_z = (\Sigma_z, \omega_z, h_z), z = 1, 2$, *constructed for* $\Gamma_z, z = 1, 2$, *and satisfying conditions 1, 2 and 3 of Theorem 1.*

Corollary 3 implies that in practice it is decidable for two P2P networks whether they function in the same manner concerning the dynamics of information.

3.3 Security in P2P Networks

Our goal is to incorporate security requirements in the formal language theoretic model of Section 3.2. We show how our model can ensure the satisfaction of integrity requirements and handle attacks against availability. We demonstrate our results via examples, e.g. the detection and elimination of a special denial–of–service attack, the SYN flooding attack against the TCP/IP handshake protocol. Our approach allows quick and efficient local analysis of security requirements, thus reducing the need for global verification.

3.3.1 TCP/IP Handshake

Let us start with a brief overview of the TCP/IP handshake protocol and its representation in our model. Let us assume that the TCP/IP handshake occurs between peers of the same

peer group, the other case can be treated in a similar manner.

The TCP/IP handshake consists of three phases (see Figure 3.1). We use grammatical rules to describe these phases as follows: $S_0 \rightarrow S_1$ (S_0 is the so–called start symbol, it reflects the fact that the given peer has initiated a TCP/IP connection, S_1 stands for message SYN), $S_1 \rightarrow S_1'$ (S_1' corresponds to message SYN–ACK), and $S_1' \rightarrow A_1$ (the establishment of the TCP/IP connection has acknowledged, A_1 denotes message ACK).

Let $\Gamma = (V, (t_1, \Theta_1, \Xi_1), \ldots, (t_n, \Theta_n, \Xi_n))$, $n \geq 1$, denote the $\mathrm{T}_{\mathrm{cX iY}}\mathrm{NPMP}_{\mathrm{F0L}}$ system, where $X, Y \in \{reg, rc\}$. Suppose that the TCP/IP connection is established between the k–th and the l–th component of the i–th team, $1 \leq i \leq n, 1 \leq k \neq l \leq r_i$, and that the components perform the individual filtering mechanism (i.e. we impose restrictions on their individual filters). Then $t_i = \{c_{i,1}, \ldots, c_{i,r_i}\}, 1 \leq i \leq n, r_i \geq 1$. Let $c_{i,k} = (P_{i,k}, F_{i,k}, \Psi_{i,k}, \Upsilon_{i,k})$ and $c_{i,l} = (P_{i,l}, F_{i,l}, \Psi_{i,l}, \Upsilon_{i,l})$, where

$$
\begin{aligned}
&P_{i,k} = \{S_0^{i,k} \rightarrow S_1^{i,k}, S_1^{i,k} \rightarrow S_1^{i,k}, S_1^{i,l'} \rightarrow A_1^{i,k}, A_1^{i,k} \rightarrow A_1^{i,k}, S_1^{i,l} \rightarrow S_1^{i,k'}, S_1^{i,k'} \rightarrow S_1^{i,k'}\},\\
&P_{i,l} = \{S_0^{i,l} \rightarrow S_1^{i,l}, S_1^{i,l} \rightarrow S_1^{i,l}, S_1^{i,k'} \rightarrow A_1^{i,l}, A_1^{i,l} \rightarrow A_1^{i,l}, S_1^{i,k} \rightarrow S_1^{i,l'}, S_1^{i,l'} \rightarrow S_1^{i,l'}\},\\
&F_{i,k} = \{S_0^{i,k}\}, F_{i,l} = \{S_0^{i,l}\},\\
&\Psi_{i,k} = \{\psi_{i,k_1}, \ldots, \psi_{i,k_{s_{i,k}}}\}, \Upsilon_{i,k} = \{\upsilon_{i,k_1}, \ldots, \upsilon_{i,k_{o_{i,k}}}\},\\
&\Psi_{i,l} = \{\psi_{i,l_1}, \ldots, \psi_{i,l_{s_{i,l}}}\}, \Upsilon_{i,l} = \{\upsilon_{i,l_1}, \ldots, \upsilon_{i,l_{o_{i,l}}}\},
\end{aligned}
$$

and $\psi_{i,k_h}, \upsilon_{i,k_j}, \psi_{i,l_{h'}}, \upsilon_{i,l_{j'}}$ are context conditions over V^*, $1 \leq i \leq n, 1 \leq k \neq l \leq r_i$, $1 \leq h \leq s_{i,k}, 1 \leq j \leq o_{i,k}, 1 \leq h' \leq s_{i,l}, 1 \leq j' \leq o_{i,l}$.

Using the above formalism, the establishment of a TCP/IP handshake (in case $c_{i,k}$ has initiated it) can be described as follows:

1. $c_{i,k}$ sends message SYN to $c_{i,l}$:
   (a) rewriting step ($c_{i,k}$ rewrites $S_0^{i,k}$ to $S_1^{i,k}$): $S_0^{i,k} \rightarrow S_1^{i,k}$,
   (b) communication step ($c_{i,l}$ receives $S_1^{i,k}$): there exist q and q', $1 \leq q \leq s_{i,k}, 1 \leq q' \leq o_{i,l}$, such that $\psi_{i,k_q}(S_1^{i,k}) = true$, $\upsilon_{i,l_{q'}}(S_1^{i,k}) = true$;

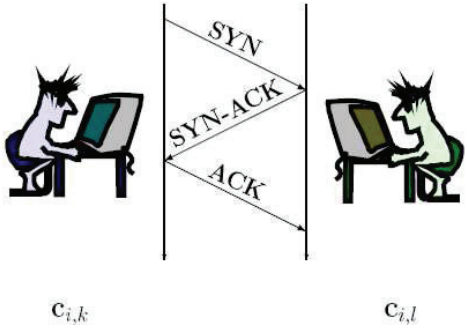

Figure 3.1: **TCP/IP handshake.** The TCP/IP handshake consists of three phases: the sending and the receipt of messages SYN, SYN-ACK and ACK.

2. $c_{i,l}$ replies back to $c_{i,k}$ with a SYN–ACK message:
   (a) rewriting step ($c_{i,l}$ rewrites $S_1^{i,k}$ to $S_1^{i,l'}$): $S_1^{i,k} \rightarrow S_1^{i,l'}$,
   (b) communication step ($c_{i,k}$ receives $S_1^{i,l'}$): there exist r and r', $1 \leq r \leq s_{i,l}, 1 \leq r' \leq o_{i,k}$, such that $\psi_{i,l_r}(S_1^{i,l'}) = true$, $\upsilon_{i,k_{r'}}(S_1^{i,l'}) = true$;
3. $c_{i,k}$ sends message ACK to $c_{i,l}$:
   (a) rewriting step ($c_{i,k}$ rewrites $S_1^{i,l'}$ to $A_1^{i,k}$): $S_1^{i,l'} \rightarrow A_1^{i,k}$,
   (b) communication step ($c_{i,l}$ receives $A_1^{i,k}$): there exist p and p', $1 \leq p \leq s_{i,k}, 1 \leq p' \leq o_{i,l}$, such that $\psi_{i,k_p}(A_1^{i,k}) = true$, $\upsilon_{i,l_{p'}}(A_1^{i,k}) = true$.

Notice that we have not detailed the configuration transmissions of the whole network.

3.3.2 SYN Flooding Attack

Now consider the SYN flooding attack against the TCP/IP handshake. We investigate two different scenarios: in the first scenario the perpetrator uses his/her real IP address in the malicious communication, in the second scenario the messages contain spoofed IP addresses.

First Scenario

Assume that the attacker p_k with IP address p_k sends N ($N \geq 1$) SYN messages to p_l. As a response to each SYN message, p_l issues a SYN–ACK message and waits for the corresponding ACK message. Since p_k is a corrupt peer, it will not send any acknowledgement to p_l. Our model detects and terminates connections with peers that has initiated the TCP/IP handshake but will not acknowledge its establishment.

Let $\Gamma = (V, (t_1, \Theta_1, \Xi_1), \ldots, (t_n, \Theta_n, \Xi_n))$, $n \geq 1$, denote the $\mathrm{T}_{\mathrm{cXiY}}\mathrm{NPMP}_{\mathrm{F0L}}$ system, $X, Y \in \{reg, rc\}$. Suppose that the handshake is established between the k–th and the l–th component of the i–th team (the case when members belong to two different peer groups may be done analogously, thus it is left to the reader), and that the components utilize their individual filters to transmit information. We refer to the nodes of the network as components instead of peers, i.e. p_k will be referred to as $c_{i,k}$ and p_l as $c_{i,l}$, $1 \leq i \leq n, 1 \leq k \neq l \leq r_i$. Then $t_i = \{c_{i,1}, \ldots, c_{i,r_i}\}, 1 \leq i \leq n, r_i \geq 1$. Let $c_{i,k} = (P_{i,k}, F_{i,k}, \Psi_{i,k}, \Upsilon_{i,k})$ and $c_{i,l} = (P_{i,l}, F_{i,l}, \Psi_{i,l}, \Upsilon_{i,l}), 1 \leq i \leq n, 1 \leq k \neq l \leq r_i$. $c_{i,k}$ can either be a honest or the malicious peer, $1 \leq i \leq n, 1 \leq k \neq l \leq r_i$. If $c_{i,k}$ is a honest peer, then its rules are the same as it is described in the case of the TCP/IP handshake. If $c_{i,k}$ is a malicious peer (we assume that $c_{i,l}$ is not malicious), then

$$P_{i,k} = \{S_0^{i,k} \to S_1^{i,k}, S_1^{i,k} \to S_1^{i,k}, S_1^{i,l'} \to B, B \to B, S_1^{i,l} \to S_1^{i,k'}, S_1^{i,k'} \to S_1^{i,k'} \mid B \in V\},$$
$$P_{i,l} = \{S_0^{i,l} \to S_1^{i,l}, S_1^{i,l} \to S_1^{i,l}, S_1^{i,k'} \to A_1^{i,l}, A_1^{i,l} \to A_1^{i,l}, S_1^{i,k} \to S_1^{i,l'}, S_1^{i,l'} \to S_1^{i,l'}\},$$
$$F_{i,k} = \{S_0^{i,k}\}, F_{i,l} = \{S_0^{i,l}\},$$
$$\Psi_{i,k} = \{\psi_{i,k_1}, \ldots, \psi_{i,k_{s_{i,k}}}\}, \Upsilon_{i,k} = \{\upsilon_{i,k_1}, \ldots, \upsilon_{i,k_{o_{i,k}}}\},$$
$$\Psi_{i,l} = \{\psi_{i,l_1}, \ldots, \psi_{i,l_{s_{i,l}}}\}, \Upsilon_{i,l} = \{\upsilon_{i,l_1}, \ldots, \upsilon_{i,l_{o_{i,l}}}\},$$

where ψ_{i,k_h}, $\upsilon_{i,k_j}, \psi_{i,l_{h'}}$ and $\upsilon_{i,l_{j'}}$ are context conditions over V^*, $1 \leq i \leq n, 1 \leq k \neq l \leq r_i$, $1 \leq h \leq s_{i,k}, 1 \leq j \leq o_{i,k}, 1 \leq h' \leq s_{i,l}, 1 \leq j' \leq o_{i,l}$,. $B \in V$ can either be the acknowledgement or an other arbitrary symbol from the alphabet.

The first and the second configuration transmissions are the same as in the case of TCP/IP handshake when $c_{i,k}$ is the attacker and $c_{i,l}$ is the victim.

In the first step of the third configuration transmission each node p_k (each component $c_{i,k}$) is presumed to perform some kind of rewriting (it can be, for instance, the identical rewriting). During the second step of the third stage $c_{i,l}$ expects a message that contains $A_1^{i,k}$ for some k (permitting context) and refuses all other messages. Then for all $B \in V \setminus \{A_1^{i,k}\}$ and for all p, p', $1 \leq p \leq s_{i,k}, 1 \leq p' \leq o_{i,l}$: $\psi_{i,k_p}(B) = true$, $\upsilon_{i,l_{p'}}(B) = false$ (the communication is between $c_{i,k}$ and $c_{i,l}$); for $A_1^{i,k}$ and for all q, q', $1 \leq q \leq s_{i,k}, 1 \leq q' \leq o_{i,l}$: $\psi_{i,k_q}(A_1^{i,k}) = false$ and $\upsilon_{i,l_{q'}}(A_1^{i,k})$ can be either $true$ or $false$. This scenario is depicted in Fig. 3.2.

Observe that we have not given a detailed description of all the configuration transmissions of the network.

We also introduce a counter that increases by one each time a malicious connection is initiated by the underlying peer. If this counter reaches a threshold N, then the honest peer refuses to accept any further communication with the malicious peer. The use of the counter allows us to terminate malicious peers in a timely manner.

To monitor the corruptness of components $c_{i,k}$, $c_{i,l}$ or the delay in the communication process, we need to check multisets $M_{i,k}^{(t)}$ and $M_{i,l}^{(t)}$, $t = 2k', k' \geq 1$. Suppose that at step t, $t = 2k' - 1, k' \geq 1$, $c_{i,k}$ has initiated a TCP/IP handshake with $c_{i,l}$. Then message SYN

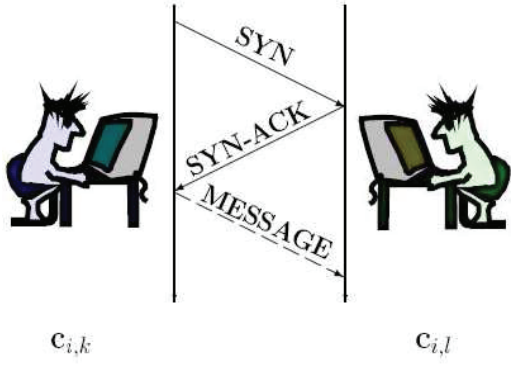

Figure 3.2: **SYN flooding attack: first scenario.** The attacker p_k (component $c_{i,k}$) with IP address p_k sends N ($N \geq 1$) SYN messages to p_l (component $c_{i,l}$). As a response to each SYN message, p_l issues a SYN–ACK message and waits for the corresponding ACK message. Since p_k is a corrupt peer, it will not send any acknowledgement to p_l.

from $c_{i,k}$ will be added to $M_{i,l}^{(t+1)}$. In this case, message SYN–ACK from $c_{i,l}$ should be added to $M_{i,k}^{(t+3)}$ and message ACK from $c_{i,k}$ to $M_{i,l}^{(t+5)}$. The counter of delay or corruptness of $c_{i,l}$ will be increased by one in case $c_{i,l}$ does not reply back to $c_{i,k}$ with message SYN–ACK at step $t+2$. If the communication occurs in a correct manner at time $t+2$, but not at time $t+4$, then the counter of delay or corruptness of $c_{i,k}$ will be increased by one.

The second peer receives either an unexpected message (which it refuses), nothing or the acknowledgement. In the first two cases the counter should be increased by one. This scenario is more general than the second one.

Second Scenario

Suppose that peer p_1 sends a SYN($\bar{p}$) message to peer p_2, where $\bar{p}$ is a spoofed IP address. In reply to message SYN($\bar{p}$), peer p_2 sends message SYN–ACK to $\bar{p}$, which in turn issues an ERROR message.

Assume again that the peers belong to the same peer group and employ the individual filtering mechanism. In this scenario, p_1 will be referred to as $c_{i,k}$, p_2 as $c_{i,l}$ and $\bar{p}$ as $c_{i,m}$, $1 \leq i \leq n, 1 \leq k \neq l \neq m \leq r_i$.

Let $\Gamma = (V, (t_1, \Theta_1, \Xi_1), \ldots, (t_n, \Theta_n, \Xi_n))$, $n \geq 1$, denote the $\mathrm{T}_{\mathrm{cx\,iy}}\mathrm{NPMP}_{\mathrm{F0L}}$, $X, Y \in \{reg, rc\}$, system that can describe the second scenario. Then $t_i = \{c_{i,1}, \ldots, c_{i,r_i}\}, 1 \leq i \leq n, r_i \geq 1$. Let $c_{i,k} = (P_{i,k}, F_{i,k}, \Psi_{i,k}, \Upsilon_{i,k})$, $c_{i,l} = (P_{i,l}, F_{i,l}, \Psi_{i,l}, \Upsilon_{i,l})$, $c_{i,m} = (P_{i,m}, F_{i,m}, \Psi_{i,m}, \Upsilon_{i,m})$, where

$$
\begin{aligned}
P_{i,k} &= \{S_0^{i,k} \to S_1^{i,\bar{m}}, S_1^{i,\bar{m}} \to S_1^{i,\bar{m}}\},\\
P_{i,l} &= \{S_0^{i,l} \to S_1^{i,l}, S_1^{i,l} \to S_1^{i,l}, S_1^{i,m'} \to A_1^{i,l}, A_1^{i,l} \to A_1^{i,l}\} \cup\\
&\qquad \{S_1^{i,m} \to S_1^{i,l'}, S_1^{i,l'} \to S_1^{i,l'}, S_1^{i,\bar{m}} \to S_1^{i,\bar{m}'}, S_1^{i,\bar{m}'} \to S_1^{i,\bar{m}'}\},\\
P_{i,m} &= \{S_0^{i,m} \to S_1^{i,m}, S_1^{i,m} \to S_1^{i,m}, S_1^{i,l'} \to A_1^{i,m}, A_1^{i,m} \to A_1^{i,m}\} \cup\\
&\qquad \{S_1^{i,l} \to S_1^{i,m'}, S_1^{i,m'} \to S_1^{i,m'}, S_1^{i,\bar{m}'} \to E, E \to E\},\\
F_{i,k} &= \{S_0^{i,k}\}, F_{i,l} = \{S_0^{i,l}\}, F_{i,m} = \{S_0^{i,m}\},\\
\Psi_{i,k} &= \{\psi_{i,k_1}, \ldots, \psi_{i,k_{s_{i,k}}}\}, \Upsilon_{i,k} = \{\upsilon_{i,k_1}, \ldots, \upsilon_{i,k_{o_{i,k}}}\},\\
\Psi_{i,l} &= \{\psi_{i,l_1}, \ldots, \psi_{i,l_{s_{i,l}}}\}, \Upsilon_{i,l} = \{\upsilon_{i,l_1}, \ldots, \upsilon_{i,l_{o_{i,l}}}\},\\
\Psi_{i,m} &= \{\psi_{i,m_1}, \ldots, \psi_{i,m_{s_{i,m}}}\}, \Upsilon_{i,m} = \{\upsilon_{i,m_1}, \ldots, \upsilon_{i,m_{o_{i,m}}}\},
\end{aligned}
$$

and $\psi_{i,k_h}, \upsilon_{i,k_j}, \psi_{i,l_{h'}}, \upsilon_{i,l_{j'}}, \psi_{i,m_{h''}}, \upsilon_{i,m_{j''}}$ are context conditions over V^*, $1 \leq i \leq n, 1 \leq k \neq l \neq m \leq r_i$, $1 \leq h \leq s_{i,k}, 1 \leq j \leq o_{i,k}, 1 \leq h' \leq s_{i,l}, 1 \leq j' \leq o_{i,l}, 1 \leq h'' \leq s_{i,m}, 1 \leq j'' \leq o_{i,m}$. We have supposed that $c_{i,k}$ is a malicious peer, $c_{i,l}$ and $c_{i,m}$ are honest peers.

The second scenario of SYN flooding attack (in case $c_{i,k}$ has initiated it) can be described as follows:

1. $c_{i,k}$ sends message SYN to $c_{i,l}$ with a spoofed IP address $\bar{m}$:
   (a) rewriting step ($c_{i,k}$ rewrites $S_0^{i,k}$ to $S_1^{i,\bar{m}}$): $S_0^{i,k} \rightarrow S_1^{i,\bar{m}}$,
   (b) communication step ($c_{i,l}$ receives $S_1^{i,\bar{m}}$): there exist q and q', $1 \leq q \leq s_{i,k}, 1 \leq q' \leq o_{i,l}$, such that $\psi_{i,k_q}(S_1^{i,\bar{m}}) = true$, $\upsilon_{i,l_{q'}}(S_1^{i,\bar{m}}) = true$;
2. $c_{i,l}$ sends message SYN–ACK to $c_{i,m}$:
   (a) rewriting step ($c_{i,l}$ rewrites $S_1^{i,\bar{m}}$ to $S_1^{i,\bar{m}'}$): $S_1^{i,\bar{m}} \rightarrow S_1^{i,\bar{m}'}$,
   (b) communication step ($c_{i,m}$ receives $S_1^{i,\bar{m}'}$): there exist r and r', $1 \leq r \leq s_{i,l}, 1 \leq r' \leq o_{i,m}$, such that $\psi_{i,l_r}(S_1^{i,\bar{m}'}) = true$, $\upsilon_{i,m_{r'}}(S_1^{i,\bar{m}'}) = true$;
3. $c_{i,m}$ replies back to $c_{i,l}$ with the ERROR message:
   (a) rewriting step ($c_{i,m}$ rewrites $S_1^{i,\bar{m}'}$ to E): $S_1^{i,\bar{m}'} \rightarrow E$,
   (b) communication step ($c_{i,l}$ receives E): there exist p and p', $1 \leq p \leq s_{i,m}, 1 \leq p' \leq o_{i,l}$, such that $\psi_{i,m_p}(E) = true$, $\upsilon_{i,l_{p'}}(E) = true$.

Figure 3.3 illustrates this scenario. Note that we have not detailed the configuration transmissions of the whole network.

To detect the SYN flooding attack, we have to investigate multisets $M_{i,l}^{(t)}$ and $M_{i,m}^{(t)}$, $t = 2k', k' \geq 1$. In this scenario, the originator of message SYN will not be known, since the IP address is spoofed. Let us suppose that some component has sent a SYN message with a spoofed IP address (denote this by $\bar{m}$) to component $c_{i,l}$ at step t, $t = 2k' - 1, k' \geq 1$. Then $\text{SYN}(\bar{m}) \in M_{i,\bar{m}}^{(t)}$ will be added to multiset $M_{i,l}^{(t+1)}$. In the second step of the next configuration transmission, $c_{i,l}$ issues a SYN–ACK message to $c_{i,m}$, because $c_{i,l}$ believes that it has received a SYN message from $c_{i,m}$ previously. Since $c_{i,m}$ is aware of the fact that it has not sent a SYN message to $c_{i,l}$, $c_{i,m}$ responds back with the ERROR message to $c_{i,l}$ (message $\text{SYN} \in M_{i,m}^{(t-1)}$ will not be added to $M_{i,l}^{(t+1)}$, message $\text{SYN–ACK} \in M_{i,l}^{(t+1)}$ and message $\text{ERROR} \in M_{i,m}^{(t+3)}$, however, will be added to $M_{i,m}^{(t+3)}$ and $M_{i,l}^{(t+5)}$, respectively).

Note that the second scenario is the special case of the first one, since issuing the ERROR message corresponds to the transmission of a non–expected message. The main advantage of this scenario over the previous one is the determinism. Furthermore, with this approach

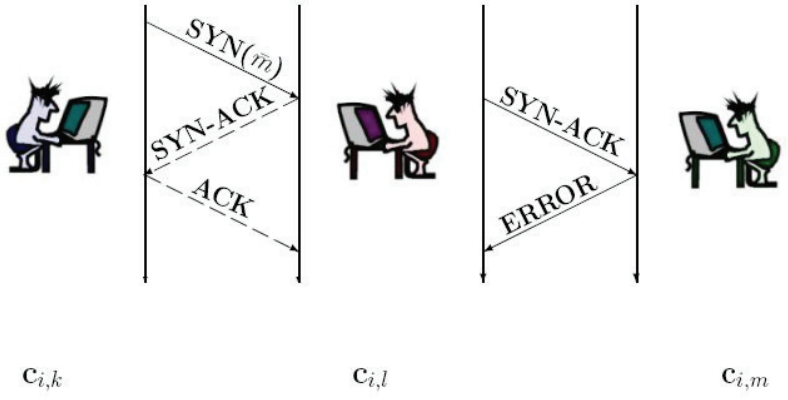

Figure 3.3: **SYN flooding attack: second scenario.** Peer p_1 (component $c_{i,k}$) sends a SYN($\bar{p}$) (SYN($\bar{m}$)) message to peer p_2 (component $c_{i,l}$), where $\bar{p}$ ($\bar{m}$) is a spoofed IP address. In reply to message SYN($\bar{p}$) (SYN($\bar{m}$)), peer p_2 sends message SYN–ACK to $\bar{p}$ (component $c_{i,m}$), which in turn issues an ERROR message.

the SYN flooding attack can be handled immediately. However, the limitation of our approach is that if $\bar{p}$ has to respond to an unexpected message, then this requirement may result in exhausting the resources of $\bar{p}$.

3.3.3 Access Control

Our model can enforce simple access control requirements. We can define for each peer the strings that are permitted to be sent and to be received, therefore our filters can be utilized to limit traffic flow. Herein we show how to employ our model to support Discretionary Access Control (DAC) [113] via filters. The DAC can be described by a

tuple: $(subject, object, \pm access\_mode)$, where *subject* is the active entity permitted (denied) access to or provides an other entity with access to a resource *object* in the mode *access_mode*. We propose the use of the following notation: $(peer, string, \pm < direction >)$ to express DAC information flow requirements. *peer* corresponds to *subject*, *string* to *object*, and $\pm < direction >$ defines whether the string is permitted or denied to enter (in) or leave (out) a filter of a peer. For instance, if in the P2P network $(peer, string, +in)$ and $(peer, string, +out)$ hold, it means that the sender is able to transmit a string, which can be either a message or an advertisement, to the receiver. The denial $(peer, string, -in)$ does not let the string in and $(peer, string, -out)$ does not let it out, hence preventing potential malicious attacks and keeping the string confidential, respectively.

Let $\Gamma = (V, (t_1, \Theta_1, \Xi_1), \ldots, (t_n, \Theta_n, \Xi_n))$, $n \geq 1$, denote the $\mathrm{T}_{\mathrm{cXiY}}\mathrm{NPMP}_{\mathrm{F0L}}$ system, where $X, Y \in \{reg, rc\}$. Suppose that the connection is established between the k–th and the l–th component of the i–th team at individual information filtering level, $1 \leq i \leq n, 1 \leq k \neq l \leq r_i$. Then $t_i = \{c_{i,1}, \ldots, c_{i,r_i}\}, 1 \leq i \leq n, r_i \geq 1$. Let $c_{i,k} = (P_{i,k}, F_{i,k}, \Psi_{i,k}, \Upsilon_{i,k})$, and $c_{i,l} = (P_{i,l}, F_{i,l}, \Psi_{i,l}, \Upsilon_{i,l}), 1 \leq i \leq n, 1 \leq k \neq l \leq r_i$. $\Psi_{i,k} = \{\psi_{i,k_1}, \ldots, \psi_{i,k_{s_{i,k}}}\}$, $\Upsilon_{i,k} = \{\upsilon_{i,k_1}, \ldots, \upsilon_{i,k_{o_{i,k}}}\}, \Psi_{i,l} = \{\psi_{i,l_1}, \ldots, \psi_{i,l_{s_{i,l}}}\}$, $\Upsilon_{i,l} = \{\upsilon_{i,l_1}, \ldots, \upsilon_{i,l_{o_{i,l}}}\}$, where $\psi_{i,k_h}, \upsilon_{i,k_j}, \psi_{i,l_{h'}}, \upsilon_{i,l_{j'}}$ are contexts conditions over V^*, $1 \leq i \leq n, 1 \leq k \neq l \leq r_i$, $1 \leq h \leq s_{i,k}, 1 \leq j \leq o_{i,k}, 1 \leq h' \leq s_{i,l}, 1 \leq j' \leq o_{i,l}$. Let us assume that the string to be communicated is denoted by ω. Then the four variants of the DAC can be described as follows:

1. there exists j, $1 \leq j \leq o_{i,k}$, such that $(c_{i,k}, \omega, \upsilon_{i,k_j}(\omega) = true)$,
2. there exists h', $1 \leq h' \leq s_{i,l}$, such that $(c_{i,l}, \omega, \psi_{i,l_{h'}}(\omega) = true)$,
3. for all j, $1 \leq j \leq o_{i,k}$, $(c_{i,k}, \omega, \upsilon_{i,k_j}(\omega) = false)$ holds,
4. for all h', $1 \leq h' \leq s_{i,l}$, $(c_{i,l}, \omega, \psi_{i,l_{h'}}(\omega) = false)$ holds.

In the first case, the communicated string is allowed to penetrate an output filter of the sender, in the second case, an input filter of the receiver, whilst in the third case, the string is not allowed to penetrate any of the output filters of the sender, and in the fourth case, any of the input filters of the receiver.

Let us consider the case when the connection is established between two peers of two (not necessarily different teams). Let us assume that component x of the i–th team engages in communication with component y of the j–th team at collective information filtering level, $1 \leq i, j \leq n$, $1 \leq x \leq r_i, 1 \leq y \leq r_j$, $x \neq y$. Let $t_i = \{c_{i,1}, \ldots, c_{i,r_i}\}, t_j = \{c_{j,1}, \ldots, c_{j,r_j}\}$, $1 \leq i, j \leq n, r_i, r_j \geq 1$. Let $\Theta_i = \{\theta_{i1}, \ldots, \theta_{ip_i}\}$, $\Xi_i = \{\xi_{i1}, \ldots, \xi_{iq_i}\}$, $\Theta_j = \{\theta_{j1}, \ldots, \theta_{jp_j}\}$, $\Xi_j = \{\xi_{j1}, \ldots, \xi_{jq_j}\}$, where $\theta_{ik}, \xi_{il}, \theta_{ju}, \xi_{jv}$, are context conditions over V^*, $1 \leq i, j \leq n$, $1 \leq k \leq p_i, 1 \leq l \leq q_i$, $1 \leq u \leq p_j, 1 \leq v \leq q_j$. Then the four variants of the DAC can be described as follows:

1. there exists k, $1 \leq k \leq p_i$, such that $(c_{i,x}, \omega, \theta_{ik}(\omega) = true)$,
2. there exists h', $1 \leq h' \leq q_j$, such that $(c_{j,y}, \omega, \xi_{jh'}(\omega) = true)$,
3. for all k, $1 \leq k \leq p_i$, $(c_{i,x}, \omega, \theta_{ik}(\omega) = false)$ holds,
4. for all h', $1 \leq h' \leq q_j$, $(c_{j,y}, \omega, \xi_{jh'}(\omega) = false)$ holds.

In the first case, the communicated string is allowed to penetrate an output filter of the sender's team, in the second case, an input filter of the receiver's team, whilst in the third case, the string is not allowed to penetrate any of the output filters of the sender's team, and in the fourth case, any of the input filters of the receiver's team.

3.4 Discussion

Peer–to–peer (P2P) networking is a rapidly growing domain of computer science but with only a few theoretical considerations. Thus theoretical foundations are justifiable by all means. The aim of this chapter is to adopt a different formal approach to the issues of information dynamics and security in P2P networks.

3.4.1 Related Work

In the sequel, we are going to give an overview of the state–of–the–art research in the field of P2P computing, focusing mainly on potential further generalizations of the formal model used for the description of P2P networks.

A relevant concept in P2P networks is self–organization. De Meer and Koppen claim that self–organization includes complexity, feedback, emergence, criticality, heterarchy, stigmergy or perturbation in the context of P2P networks (see [120], Chapter 15), whereas Aberer et al. (see [120], Chapter 10) assert that it resides in the distribution of control, the locality of processing and the emergence of global structures from local interactions. In our formal language theoretic model self–organization can be viewed as the emergence of the intensive interaction of a multiset string processor with its dynamic environment, i.e. a collection of separated sub–environments, each belonging to an other multiset string processor.

Web service technology provides functionality over the Internet that can be accessed through well–defined interfaces (see [120], Chapter 14). Hillenbrand and Müller propose some design goals that have been adopted to either technology and might prove beneficial in the other. Nonetheless, the combination of P2P and web service technology poses some serious issues such as security to be overcome. Through a well–defined input filter system we can enforce access control and restrict availability of resources to guarantee security and defend the network against undesirable effects in our mathematical model.

It is demonstrated that P2P networks bear a close resemblance to the ubiquitous computing world (see [120], Chapter 27) with regard to ad hoc communication, feasibility of wireless network structure, rapid information flow, collaboration and resource sharing. Thus it can be deduced that the mathematical model presented in this chapter surpasses the formalization of P2P networks.

Distributed information retrieval is of prime importance in P2P networks [93]. To this end, we introduce some kind of hierarchy into the network. We maintain the list of faulty and faultless peer. We apply the constraints of the grammar to detect faulty messages. Local satisfaction of these constraints guarantees the global testing of the system. Testing

is restricted to testing of a peer and peers engage in communication with the given peer, thus combinatorial tests can be avoided. Unlike [93], we do not assume strong cooperation among all the peers herein.

Kant et al. propose a taxonomy for the classification of P2P technologies [65]. The taxonomy reveals several research issues to be explored, such as friendliness, security and access control. The concept a friendliness in a hostile environment can be realized through the introduction of apprentice peers aiding the peers in determining with whom it is worth communicating. It is possible to elaborate more expressive policy languages than DAC in our model. The specification of both positive and negative authorizations and the notions of authorization derivation, conflict resolution and various decision strategies should be integrated. Different strategies could be applied to different users, groups, objects, or roles, based on the needs of the security policy. In this way the expressive policy neutral Flexible Authorization Framework (FAF) [61], [131] can be formulated.

In [64] Kant and Iyer study the evolution of P2P communities in terms of the path of the response, reachability, abandonment or retry of a query and the hybrid nature of the P2P architecture. In our model the path of the reply can be determined through the communication functions. The various communication functions describe the reachablility of the peers and their contribution to the community, i.e. the satisfaction of the demands of other peers, concerning the requested information, either at collective, at individual or at both levels. The communication functions guarantee that it is possible to realize load balancing in the model, i.e. the support of intelligent redistribution based on the access frequency and the location of the content [64]. As a consequence, a minority of nodes providing the queriers with information may be prevented from becoming hot–spots [64]. The abandonment or the retry of a query may exhaust the resources of the information provider, which can pose some security requirements in P2P networks to be dealt with.

P2P systems are vulnerable to network protocol–based attacks, including denial–of–service attacks. Giuli et al. elaborate a defensive framework against malicious adversaries in the form of filters [52]. In our work, it is also the filter system that protects the network against perpetrators. Owing to the constraints imposed on the grammars, it is not needed to have the votes of a quorum as it does in [52], through local satisfaction of the formal language

theoretic constraints, the malicious peers can be detected, and as a result, combinatorial tests can be avoided.

3.4.2 Main Achievements and Further Considerations

In this chapter we have equipped the networks of parallel multiset string processors with teams of collective filtering introduced in [28] with the individual filtering mechanism. The individual filtering mechanism makes it possible for peers to use a hybrid filtering method, since the collective and individual filters can be of different types (for instance, the access to the resources of a P2P system depends on whether the collective or the individual filtering mechanism is employed).

Another generalization of the model described in [28] is that there is no need to apply a static neighbourhood relation, a multiset string processor is allowed to communicate with another, in case certain conditions are satisfied, and as a consequence, the relationship between two multiset string processors varies dynamically, as in the case of wireless networks. Context conditions are given in the form of filters. Should the communicated string be a message, then the output filter (either at the level of the multiset string processor or at that of the team the given multiset string processor belongs to) can be regarded as the output pipe, and the input filter as the input pipe. In this way, pipes can be realized in the networks of parallel multiset string processors with teams of collective and individual filtering.

Both the output and the input pipe can be only in two states, namely they can be off or on. Should an output pipe of a given multiset string processor be off (which corresponds to the fact that the underlying context condition is not satisfied), then the multiset string processor is not be able to send a message. If an input pipe of the multiset string processor is on (which corresponds to the fact that the underlying context condition is satisfied), then it is able to send the previously compiled message with a time limit. The receiver may accept any message within that time limit. The satisfiability of the time limit can be guaranteed by the application of context conditions encoding it (the use of a disjoint alphabet to describe the time limit has no influence on the mathematical results demonstrated in this chapter,

therefore it is omitted).

In theory, a well–defined input filter system can protect the network against overloading and faults stemming from the function of the system. In P2P systems different types of faults can occur. The information sent to other peers may be erroneous, lacking, or present in an undesirable amount. An additional problem may arise in case the required information does not arrive in due course. The faults that emerge during the function of the network may substantially influence how such a system works. Faults may superpose each other regardless of how high percentage of the nodes of the network function in a correct manner. It is essential to maintain the filters in the appropriate state in order to protect the system against undesirable effects. It can be assumed that each peer has some tool to determine whether the message received from a given peer is defective or not, which should enable the peer to decrease the frequency of the communication with the malicious peer or terminate the process immediately. The question may arise how it can be realized.

One possibility is that a peer sends message to itself, which contains the list of faulty peers or its complement, the list of friends.

Another potential is that the peers are equipped with one–step buffers. For the sake of future generalizations, the peers will be called master peers, whereas the one–step buffers apprentice peers. The apprentice peer maintains the list of the faulty or faultless peers, which it sends to its master peer at each time step. Afterwards, the master peer can control its inputs and outputs subject to the list, rewritten according to the actual information received from other members of the P2P network. The introduction of a buffer does not have an influence on the results of Theorem 1, because it can be regarded as a special multiset string processor, whose rewritings are identical ones, hence the solution is viable. Indeed, a general multiset string processor is capable of doing more and this is why the master and the apprentice peer are distinguished. In effect, more than one apprentice peer may belong to a master peer, and as a consequence, a hierarchy of processors might be constructed. These concepts are elaborated below.

The maintenance of the list of peers is motivated by Internet crawler experiments (see, e.g. [101]), where the list of good URLs corresponds to the list of friends of the underlying Internet crawler, with whom it is worth 'communicating'. If the crawler happens to receive

faulty (insignificant or obsolete) information from an URL, then its list of friends is updated. The maintenance of such lists has proven to be very efficient [99] in scale–free small world networks (see, e.g. [14] and references therein). Evolving structures seem to generate scale–free small world structure.

There are several approaches in the literature that help peers to decide which members of a P2P system send them a faulty message [2, 17, 82, 88, 124, 133]. Collaboration may be facilitated by the selection of trustworthy partners. Obviously, the identification and the exclusion of malicious and egoistic peers, and the rehabilitation of reliable ones, contribute enormously to the efficiency of the network.

Another application of the apprentice peer is that it introduces memory into the system and enables feedback. Using the terminology of neural networks [54], let the experienced input at time t be denoted by $x_e(t)$. For input $x_e(t)$ the system produces the output $y(t)$. The output may serve some goals and in general, the system may have *expectations* of future inputs. If the expectations are not fulfilled, then the system may generate correcting outputs. This type of functioning is enabled by the apprentice peer. The master peer may derive the output as well as its desired next input $x_d(t+1)$ in reply to the input $x_e(t)$ at time t. The output can be sent to other peers, whilst the desired next input $x_d(t+1)$ to the apprentice. The apprentice sends the output back at the next time instant. Thus, at the next time step, the master will have access both to the next experienced input $x_e(t+1)$ and the desired input $x_d(t+1)$, and it can derive correcting outputs provided that the two inputs do not coincide. This type of computation is called first order feedback scheme. Analogously, second order feedback schemes can be constructed if the one–step buffer is extended to a multistep buffer. Apprentices assist peers with their feedbacks so that error correction is realizable in P2P networks.

The concept of apprentice peers surpasses multistep buffers. It can be easily seen that pipeline systems and pipelined operations fit in with our considerations [100]: the master peer sends the input to the pipeline and the last apprentice of the hierarchy provides the other peers awaiting the result of the computation with the output of the pipeline.

Furthermore, it can be anticipated that the adaptive grammatical model of P2P networks can take a crucial part in the automation of a testing process. It is widely known that

nowadays distributed hardware systems possess all characteristics of complex systems and are difficult, sometimes impossible, to be tested. A compiler that is able to check at a higher level the logical consistency of a software as a rule–based system, for instance, as a formal language theoretic construction, does not exist.

In telecommunications most software units can be viewed as reactive systems that receive stimuli from their environment and respond to them by emitting observable output signals after their internal states have been altered [115]. As a result, it is a natural way to model such systems as finite state machines [58, 110, 112]. Nonetheless, rarely do the software units function in a way as they are expected. Unexpected inputs might emerge in lieu, which means that in this respect they behave rather like stochastic machines. An essential future direction of the formal language theoretic approach described in this chapter is that the features of the P2P network may be warranted by the grammars. At present, if the constraints imposed on the grammars are satisfied in the case of each peer, then the system does not need to be tested globally. Instead, testing can be restricted to the individual testing of the peers and as a result, combinatorial tests can be avoided.

In conclusion, owing to the grammar systems theoretic formalization of P2P systems several advantages can be gained. In particular, the concept of apprentice peer makes the P2P system flexible in certain aspects. In the first place, it can render the P2P network adaptive through the maintenance of lists and it makes task–based dynamic configuration possible. In the second place, apprentice peers can be used for computing correcting outputs, the tool of short term adaptation. Lastly, the introduction of the apprentice peers enables pipelined operations in P2P networks.

Chapter 4

Internet Crawlers in Quest of Novel Information

As we have seen in Chapter 3, the selection of reliable partners of a peer is inspired by Internet crawler experiments. A novel piece of information found by a crawler can serve as a context for another crawler and can help to accelerate search. Each crawler maintains a list of the most promising URLs, called the weblog, and updates this list periodically. Analogously, the apprentice peer keeps track of the trustworthy peers, which it communicates at each instant to the master peer. Therefore the weblog can be regarded as the list maintained by the apprentice. In the case of P2P networks, peers use regular filters for the transmission of information. In this chapter we demonstrate that besides random context conditions a whole panoply of regulated rewriting devices is at our disposal to describe the behaviour of complex systems of cooperating and communicating agents. We prove that if we ignore the aging of the web pages in the model, then systems with rather simple component grammars suffice to identify any recursively enumerable language. Whereas if the web pages may become obsolete, then the efficiency of the cooperation of the agents decreases substantially. We also examine the extent to which communication makes a goal–oriented community efficient in different graph topologies through simulations. The results of this work appeared in [72, 74, 75] and [78].

The outline of this chapter as follows. In Section 4.1, we review our problem domain in general. In Section 4.2, we deal with the literature on which our work is based. In Section 4.3, we characterize the forager architecture. In Section 4.4, we present our formal language theoretic model employed to describe the behaviour of the foragers. In Section 4.5, we summarize the results of our experiments. In Section 4.6, we argue the further research potentials from both theoretical and practical points of view.

4.1 Introduction

The World Wide Web is an exponentially growing and a dynamically changing information source. It demonstrates the scale–free small world property [15, 84, 129]. Owing to the scale–free small world nature of the World Wide Web to locate novel information often requires strenuous efforts, hence the need of the elaboration of efficient crawling algorithms.

In our framework the crawlers employ either the selective learning, the function approximation–based reinforcement learning (RL) or their combination. In our experimental study, we compare the selective learning algorithm with the linear function approximation–based reinforcement learning algorithm. The crawlers may communicate in a direct and in an indirect manner simultaneously. The indirect communication is invoked by the reward system: positive reward is delivered only to the first sender of a news item. The crawlers have to find either novel documents, novel documents within time limits, novel documents on different topics or documents satisfying some possible combination of the previous criteria. We collect real data from the Web and create different graph topologies through link reorganization: scale–free worlds (SF), scale–free small worlds (SFSW) and random world environments (RWE).

In the formal language theoretic model we concentrate only on the indirect communication. We capture the behaviour of the crawlers by a variant of eco–grammar systems, called eco–foraging systems. We claim that if we ignore the aging of the web environment in the model, then through the simulation of certain normal form grammars, the eco–foraging systems determine the class of recursively enumerable languages. Whereas if the web pages may become obsolete, then the language family generated by unordered scattered context

grammars of finite index can be obtained. The ignorance of lifetime implies that the crawlers communicating only through the environment are able to identify any computable set of the environmental states. The lifetime constraint, however, decreases the efficiency of the cooperation of the agents considerably.

4.2 Related Work

In the sequel, we review the state–of–the–art research related to the field of web crawling techniques, selective and reinforcement learning and grammar systems theory.

4.2.1 Web Crawling Techniques

Different approaches exist in the literature that discuss information retrieval on the World Wide Web. Pinkerton [106] applies breadth first, exhaustive crawlers and defines the beginning of the search area by means of anchor text of links as a potential predictor. In [22], Cho et al. employ URL ordering, a principle based on the characteristics of links, when defining the decisions of crawlers. Focused/topic–specific/topical crawlers aim at seeking and retrieving only the subset of the World Wide Web that pertains to a specific topic of relevance [37]. Focused crawling was first introduced by Chakrabarti et al. [21]. Conceptual knowledge concerning the topic plays a crucial part in a plethora of approaches to seeking novel information on the World Wide Web. In [86], [87] and [108], the authors propose the use of reinforcement learning methods so that the crawlers are able to extract relevant information while spidering the Web. Menzer et al. [89] study some machine learning issues in the case of topical crawlers and besides the role of exploration versus exploitation, they also examine the role of adaptation (learning and evolutionary algorithms) versus static approaches. In order to ameliorate the performance of the focused crawlers Diligenti et al. [37] as well as Pant and Srinivasan [103, 104] utilize popular classification methods such as Naive Bayes, Support Vector Machine (SVM) and Neural Network classification schemes [54, 81], which are well–established techniques in the areas of text and data mining.

4.2.2 Selective and Reinforcement Learning

In our experimental study we rely on selective learning and function approximation–based reinforcement learning. In selective learning alternative solutions coexist and while organisms compete for space and resources, the more efficient solutions are maintained [98]. The selection may occur, e.g. at the level of the individuals, or at the level of the solutions found by the individuals. For reviews of the evolutionary theories and the dynamics of self–modifying systems, consult [23, 49] and [63], respectively. In a typical reinforcement learning problem what motivates the learning process is the optimization of the expected value of long–term cumulated profit of the actual state (state value) or state–action pair (state–action value) [122]. A well–known RL example is Tesauro's TD–Gammon program [125], which employs MLP function approximators for value estimation. Reinforcement learning has also been utilized in concurrent multi–robot learning, see, e.g. [85], where the robots have to learn to forage in concert via direct interaction.

4.2.3 Grammar Systems Theoretic Background

In our formal language theoretic model we consider a restricted variant of eco–grammar systems (see Chapter 1 and 2), called eco–foraging systems. The changing environment represents the knowledge space to be discovered by the crawlers, i.e. the agents in the grammar systems theoretic model. The environment is given in the form of a string, the elements of the string correspond to the web pages. The itinerary of the agents, or more precisely, the next piece of information to be discovered is predefined in some way. To this end, we represent the agents as special, very simple programmed grammars [36]. These agents demonstrate a remarkable behaviour, since they are able to describe the language family generated by unordered scattered context grammars of finite index and the language family of recursively enumerable languages under different assumptions, whilst the individual components can generate finite languages only.

4.3 Forager Architecture

In the sequel, we describe the agents employed in our experiment and their working mechanisms. Two types of agents, i.e. the foragers/the crawlers and the reinforcing agent (RA) can be distinguished in our model. The terms *forager* and *crawler* can be used interchangeably[1]. The foragers crawl the web and send back the addresses (URLs, uniform resource locators) of the selected documents to the RA. The RA is a simple machine scheduling the work of the foragers: it launches the foragers consecutively for an equal number of times. In effect, it acts as a central reinforcing agent: it delivers positive reward only to the first sender of a document. Each forager visits the linked URLs in a predefined order. Indeed, regions abundant in information may be separated from each other by those lacking in information, the latter territories form *barriers*. This segregation promotes the reinforcement–based value estimation. The reinforcement–based value estimation enables the crawler to overcome short–term penalties in the hope of amassing more profit in the long run [71, 80].

4.3.1 Weblog Algorithm

Each forager possesses a *weblog* consisting of the URLs with their associated *weblog values* in descending order and recommences its activity periodically. At the beginning of a period, the forager selects randomly a URL from the best elements of the weblog. The sequence of visited URLs between two restarts forms a path.

In effect, the weblog value of a URL estimates the expected sum of rewards along the path after a visit to the underlying URL. The weblog is altered before a new path has started. The weblog value of a URL already in the weblog is modified towards the sum of rewards of the remaining part of the path, whilst that of new one is the actual sum of rewards that can be collected along the rest of the path after a visit to the given URL. The high weblog value of a URL indicates an abundance of relevant documents around it. Consequently, it

[1]The pieces of information serve as food or supplies for the crawlers, for this reason they can also be called foragers [78].

is advisable to launch a search from that URL.

Weblog Algorithm–Based Crawl

Each document is characterized by an $N = 50$ dimensional vector. The components of the vector are mapped nonlinearly onto the interval $[0, 1]$. The vector is computed by the probabilistic term–frequency inverse document–frequency (PrTFIDF) text classifier method [62], generated on a previously downloaded portion of the Internet. Every forager has a randomly chosen N dimensional *weight vector*. At a given URL, the crawler calculates the scalar products of its weight vector and all vectors belonging to the documents at the *frontier* (the list of linked but not yet visited URLs found during the crawl along the actual path). Then the crawler moves to (one of) the maximum scalar product valued URL(s).

4.3.2 Reinforcement Learning

A forager can estimate the long–term cumulated value/profit of a URL according to the reinforcements obtained after the URL has been visited. The (immediate) profit is the difference between the rewards and the penalties received at any given step. In fact, the immediate profit characterizes a step to a URL in a myopic manner. Foragers employ an adaptive linear value estimator (ALE) [122] to overcome this short sightedness. They follow the *policy* maximizing the expected long–term cumulated profit (LTP) in lieu of the immediate. The policy and the profit estimation are related concepts: the profit estimation determines the policy, whereas the policy influences the choices and, in turn, the expected LTP [122].

RL–Based Crawl

The forager performs a step according to ALE. The *weight vector* is trained by the ALE in order to improve the crawl. The LTP of the actual URL is estimated as it is described below. At a given URL, the forager computes the scalar products of its weight vector

and all vectors belonging to the documents at the frontier (the list of linked but not yet visited URLs found during the crawl). Then it greedily moves to (one of) the highest scalar product valued URL(s). After the step has been performed and the sum of the components of the immediate reward (cost of a step, cost of sending a document, rewards received for novel and topic relevant documents) has been calculated, then the error of our state value estimation δ can be computed as follows:

$$\delta = a - \gamma(b + c),$$

where a denotes the LTP of the previous URL, b the LTP of the actual URL, c the immediate reward and $0 < \gamma < 1$ the discount factor. The estimation may be regarded as perfect in case there is no error. Otherwise, the weight vector has to be modified, in proportion to the sign and the magnitude of the error. This method is called *temporal differencing* (TD). For further details about the TD method and an in–depth description of our algorithms the interested reader may consult [99] and [122], respectively. The procedure gives rise to adaptive crawl. At the beginning of each period, the forager continues the previous path.

4.3.3 The Combined Algorithm

The two algorithms, i.e. the weblog–based selective learning and the function approximation–based reinforcement learning, can be combined. The selective learning modifies the starting URL lists of the crawlers, whilst RL updates the weight vector of the crawler.

4.3.4 Sending of Documents

Crawlers apply a threshold for sending a document. The *downloaded documents* are not always novel or their values do not pass the threshold. A document will be dispatched provided that it is novel and its value passes the threshold. Neither do all the *sent documents* invoke positive rewards. If the reward delivered for a sent document is positive, then the document is a *relevant sent document*, or briefly, a *relevant document*. If the documents can be sent only to the RA, then no direct communication occurs between the crawlers.

Nonetheless, crawlers are *coupled*: they are adaptive and work in different domains and/or they tend to follow different paths [102].

The value of a document, in fact, is the scalar product of the *sending weight vector* and its PrTFIDF vector. Sending can be adaptive, for instance, if the crawlers use the PrTFIDF vector of a document and adjust the components of the sending weight vector, i.e. the sending weights, by averaging the weights of the relevant documents. In a changing world, the moving window averaging might improve performance. Two cases can be distinguished according to the estimation of the weights used for document sending: the crawlers may send the selected documents either (i) to the RA, or (ii) to another forager. In the latter case, if the crawler forwards the document to the RA, then the value of the document has to pass the sending threshold of both crawlers. This condition can be weakened provided that crawlers communicate their sending weights to their partners.

4.3.5 Topic–Specific Crawlers

Topic–specific crawlers are rewarded only for the sent documents belonging to a particular topic. In the experiments, crawlers seek either novel information, irrespective of the topic, or novel information on different topics. We define the topics by means of keyphrases. The keyphrases are determined by the keyphrase–extracting algorithm [48]. A crawler is rewarded in case the extracted keyphrase set of the sent document contains all keyphrases of a given topic. The keyphrases have been selected meticulously: at least 100 and at most 2000 documents belong to each topic in our downloaded database.

4.4 Formal Definitions

In this section we introduce the notion of eco–foraging systems (FEG systems) to model the behaviour of Internet crawlers in quest of novel information. Whilst harvesting information on the web these crawlers compete as well as cooperate/collaborate with each other. Eco–foraging systems have two main components: the web environment and the agents. We

focus on eco–foraging systems in which the components are represented as programmed grammars, or more precisely, as programmed grammar schemes.

4.4.1 Eco–Foraging Systems

First, we deal with eco–foraging systems that model the case when no lifetime is associated with the web pages, i.e. we ignore that during web crawling some pages may become obsolete. Now we define the web environment (the environment in eco–grammar systems). The web environment represents the continuously changing World Wide Web domain.

Definition 15 *The web environment with n foragers, $n \geq 1$, is a construction*

$$E = (V_E, T'_E, \mathcal{P}_E),$$

such that

- V_E *is a finite alphabet,* $V_E = V_M \cup T'_E \cup V_N \cup \bar{V}_N$, *with* $V_N = \bigcup_{i=1}^{n} N_i$ *and* $\bar{V}_N = \bigcup_{i=1}^{n} N_i^{(i)}$, *where*
  - V_M *is a finite set,*
  - $N_i = \{X_{i,1}, \ldots, X_{i,s_i}\}, N_i^{(i)} = \{X_{i,1}^{(i)}, \ldots, X_{i,s_i}^{(i)}\}$, $1 \leq s_i$, $1 \leq i \leq n$, *are finite alphabets,*
  - $T_E = \bigcup_{j=1}^{k} N_{i_j}$, *and for some* k, $1 \leq k \leq n$, $\{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}$,
  - $T'_E = \{Z' \mid Z \in T_E\}$,
  - V_M, T'_E, V_N, *and* $\bar{V}_N$, *are pairwise disjoint sets,*
- $\mathcal{P}_E = \{P_{E_1}, \ldots, P_{E_r}\}$, *where* P_{E_q}, $1 \leq q \leq r$, *is a finite set of rules of the following forms:*
  - $Y \to \alpha$, *where* $Y \in V_N, \alpha \in V_N^*$,
  - $Z^{(i)} \to \beta$, *where* $Z^{(i)} \in N_i^{(i)}$, $1 \leq i \leq n$, *and* $\beta \in V_N^* \cup V_N^* Z^{(i)} V_N^*$,

– $Z^{(j)} \to Z'$, $Z' \to Z'$, *where* $Z^{(j)} \in N_j^{(j)}$, $1 \leq j \leq n$, $N_j \subseteq T_E$, $Z' \in T'_E$ *and* $Z \in T_E$,

– $U \to \gamma$, *where* $U \in V_M$ *and* $\gamma \in V_N^* V_M^* V_N^*$.

Moreover, any rule set in $\mathcal{P}_E$ *is complete, i.e. for any* $c \in V_E$, *there is at least one rule in any* P_{E_q}, $1 \leq q \leq r$.

In Def. 15, V_E is the alphabet of the web environment, i.e. the web pages that can be altered through the joint action of the foragers and the web environment. V_E consists of the union of all alphabets N_i and $N_i^{(i)}$, $1 \leq i \leq n$, T'_E and V_M. The elements of N_i correspond to web pages that can be identified, while those of $N_i^{(i)}$ to web pages that were actually visited by the i–th forager. T_E represents the web pages that should be visited by the foragers, T'_E describes that those web pages that should be visited were really recognized by the foragers and reinforced later by the environment. The symbols from V_M characterize how the web environment works, i.e. they cannot be rewritten by any of the agents. $\mathcal{P}_E$ is the set of all rule sets P_{E_q}, $1 \leq q \leq r$, where each P_{E_q} is a set of rules (the so–called evolution rules): it describes the update of a non–visited web page, of a visited one and some other kinds of rewritings, respectively. In particular, rules of the form $Y \to \alpha$, where $Y \in V_N$, $\alpha \in V_N^*$, correspond to the update (insertion of new web page(s) into the environment, the deletion or the substitution of some part of the environmental state) of a non–visited web page, rules of the form $Z^{(i)} \to \beta$, where $Z^{(i)} \in N_i^{(i)}$, $1 \leq i \leq n$, and $\beta \in V_N^* \cup V_N^* Z^{(i)} V_N^*$, express that the actually visited web page has been deleted or left unaltered and at the same time some new web pages may have been inserted, rules of the form $Z^{(i)} \to Z'$, $Z' \to Z'$, where $Z^{(i)} \in N_i^{(i)}$, $1 \leq i \leq n$, $N_i \subseteq T_E$, $Z' \in T'_E$ and $Z \in T_E$, represent that the web pages visited by the foragers are reinforced by the environment, rules of the form $U \to \gamma$, where $U \in V_M$ and $\gamma \in V_N^* V_M^* V_N^*$, describe that symbols from the finite set V_M have been rewritten and/or some new web pages have been inserted.

We impose some constraint on the rules of the agents of the eco–grammar systems to describe the search strategy of these agents.

Definition 16 *A programmed eco–foraging system with appearance checking (an* $\mathrm{FEG}_{\mathrm{PR}_{ac}}$ *system) of degree* n, $n \geq 1$, *is a construction*

$$\Gamma = (E, A_1, \ldots, A_n, c_{init}),$$

such that

- $E = (V_E, T'_E, \mathcal{P}_E)$ *is the web environment (see Def.15),*
- $A_i = (N_i \cup N_i^{(i)}, S_i, R_i)$, $1 \leq i \leq n$, *is the i–th forager, a programmed grammar scheme with appearance checking, where*
  - $N_i \cup N_i^{(i)}$ *is the nonterminal alphabet of the i–th forager (see Def.15),*
  - $S_i \in N_i$ *is the start symbol of the i–th forager,*
  - R_i *is a finite set of triplets of the following forms:*
    - $(l_{i,1} : S_i \rightarrow S_i^{(i)}, \sigma_i(l_{i,1}), \psi_i(l_{i,1}))$, $\sigma_i(l_{i,1}) \subseteq \{l_{i,1}, \ldots, l_{i,s_i}\}$, $\psi_i(l_{i,1}) = \{l_{i,1}\}$, *is called the initial rule of the i–th forager,*
    - $(l_{i,k} : X_{i,k} \rightarrow X_{i,k}^{(i)}, \sigma_i(l_{i,k}), \psi_i(l_{i,k}))$, $X_{i,k} \in N_i \setminus \{S_i\}, X_{i,k}^{(i)} \in N_i^{(i)} \setminus \{S_i^{(i)}\}$, $2 \leq k \leq s_i$, *with* $\sigma_i(l_{i,k}) \subseteq \{l_{i,1}, \ldots, l_{i,s_i}\}, \psi_i(l_{i,k}) \subseteq \{h_{i,2}, \ldots, h_{i,s_i}\}$, *or*
    - $(h_{i,k} : X_{i,k}^{(i)} \rightarrow X_{i,k}^{(i)}, \sigma_i(h_{i,k}), \psi_i(h_{i,k}))$, $X_{i,k}^{(i)} \in N_i^{(i)} \setminus \{S_i^{(i)}\}$, $2 \leq k \leq s_i$, *with* $\sigma_i(h_{i,k}) \subseteq \{l_{i,1}, \ldots, l_{i,s_i}\}, \psi_i(h_{i,k}) \subseteq \{h_{i,2}, \ldots, h_{i,s_i}\}$, *where*
      - $Label(R_i) = \{l_{i,1}, \ldots, l_{i,s_i}, h_{i,2}, \ldots, h_{i,s_i}\}$ *is the set of labels of the rules in* R_i.
- $c_{init} = (l_{1,1}, \ldots, l_{n,1}; \omega_{init})$ *is called the initial configuration of* Γ, *where* $l_{i,1}$ *is the label of the initial rule of the i–th forager,* $1 \leq i \leq n$, $\omega_{init} = z_1 S_{j_1} z_2 \ldots z_k S_{j_k} z_{k+1}$, $S_{j_h} \in N_{j_h}, z_l \in V_E^*, 1 \leq h \leq k, 1 \leq l \leq k+1$, *and for some* k, $0 \leq k \leq n$, $\{j_1, \ldots, j_k\} \subseteq \{1, \ldots, n\}$. *The string* ω_{init} *is called the initial state of the web environment of* Γ *or the initial environmental state.*

In Def. 16, the agents or foragers are special programmed grammar schemes with appearance checking. $S_i \in N_i$ is the first web page that the i–th agent has to visit. The agents have two types of rules except for the initial step. $S_i \rightarrow S_i^{(i)}$ is the initial rule of the i–th agent. Not until the forager has visited the first web page, will it be able to jump to any of its subsequent rules. At subsequent steps, the rules of the i–th agent have the forms

$X_{i,k} \to X^{(i)}_{i,k}$, $X_{i,k} \in N_i \setminus \{S_i\}, X^{(i)}_{i,k} \in N^{(i)}_i \setminus \{S^{(i)}_i\}$, or $X^{(i)}_{i,k} \to X^{(i)}_{i,k}$, $X^{(i)}_{i,k} \in N^{(i)}_i \setminus \{S^{(i)}_i\}$, $1 \leq i \leq n$, $2 \leq k \leq s_i$. Rules $X_{i,k} \to X^{(i)}_{i,k}$, $X_{i,k} \in N_i \setminus \{S_i\}, X^{(i)}_{i,k} \in N^{(i)}_i \setminus \{S^{(i)}_i\}$, $1 \leq i \leq n$, $2 \leq k \leq s_i$, describe that the i–th agent tries to visit a not yet discovered web page. Rules $X^{(i)}_{i,k} \to X^{(i)}_{i,k}$, $X^{(i)}_{i,k} \in N^{(i)}_i \setminus \{S^{(i)}_i\}$, $1 \leq i \leq n$, $2 \leq k \leq s_i$, on the other hand, express that the i–th agent goes to a web page that it has discovered previously. As the initial state of the web environment, ω_{init} indicates, initially, we do not suppose that every agent is able to commence its work. When the agents start their work, they have to apply their initial rules.

In the sequel, we define the way in which eco–foraging systems work.

Definition 17 *Let $\Gamma = (E, A_1, \ldots, A_n, c_{init})$ be a $\mathrm{FEG}_{\mathrm{PR}_{ac}}$ system of degree n, $n \geq 1$. An $(n+1)$–tuple $c = (k_1, \ldots, k_n; \omega_E)$, where $k_i \in Label(R_i)$, $1 \leq i \leq n$, $\omega_E \in V^*_E$, is called a configuration of Γ. ω_E is the state of the web environment of Γ in configuration c or the environmental state in configuration c.*

Definition 18 *Let $\Gamma = (E, A_1, \ldots, A_n, c_{init})$ be a $\mathrm{FEG}_{\mathrm{PR}_{ac}}$ system of degree n, $n \geq 1$ (see Def.15), and let $c_1 = (k_1, \ldots, k_n; \omega_E)$ and $c_2 = (k'_1, \ldots, k'_n; \omega'_E)$ be two configurations of Γ. We say that c_1 directly derives c_2 in Γ, written as $c_1 \Longrightarrow_\Gamma c_2$, if the following conditions hold:*

1. *$\omega_E = u_1\alpha_{i_1}u_2 \ldots u_k\alpha_{i_k}u_{k+1}$ and $\omega''_E = u_1\beta_{i_1}u_2 \ldots u_k\beta_{i_k}u_{k+1}$,*
   where for some k, $0 \leq k \leq n$, $\{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}$, $\alpha_{i_j} \in N_j \cup N^{(j)}_j, \beta_{i_j} \in N^{(j)}_j$, $1 \leq j \leq k$, $u_h \in V^*_E$, $1 \leq h \leq k+1$,
2. *$(k_{i_j} : \alpha_{i_j} \to \beta_{i_j}, \sigma(k_{i_j}), \psi(k_{i_j})) \in R_{i_j}$ and $k'_{i_j} \in \sigma(k_{i_j})$, $1 \leq j \leq k$,*
3. *there is no $m \in \{1, \ldots, n\} \setminus \{i_1, \ldots, i_k\}$, such that $(k_m : \alpha_m \to \beta_m, \sigma(k_m), \psi(k_m)) \in R_m$ can be applied to $u_1u_2 \ldots u_{k+1}$,*
4. *$k'_m \in \psi(k_m)$ for $m \in \{1, \ldots, n\} \setminus \{i_1, \ldots, i_k\}$,*
5. *$\omega'_E = v_1\beta_{i_1}v_2 \ldots v_k\beta_{i_k}v_{k+1}$, where $u_1 \ldots u_{k+1} \Longrightarrow v_1 \ldots v_{k+1}$ is a $0\mathrm{L}$ rewriting according to some P_{E_q}, $1 \leq q \leq r$, $P_{E_q} \in \mathcal{P}_E$.*

In Def. 18, the programmed grammar schemes determine the next rule to be applied on the basis of the previous one(s). If the forager has managed to identify a news element, then it will try to search for a novel one. If the attempt of the forager is not successful and it has not yet commenced its work, then it will try to visit its first web page again. If the forager fails to discover a web page different from the initial one, then it will go to a web page that it has discovered previously. If the crawler has managed to identify the previously discovered web page, then it will go to a not yet visited page, otherwise to a visited one.

The next state of the web environment is determined both by the action rules of the foragers and the set of rules of the web environment. As the reader may observe, the evolution rules of the environment are applied in the same manner as a 0L system. If the environment has more than one set of evolution rules, then it behaves like a T0L system. The actions of the foragers have priority over the evolution of the web environment. The foragers have to perform their actions simultaneously.

The transitive (and reflexive) closure of $\Longrightarrow_\Gamma$ is denoted by $\Longrightarrow_\Gamma^+$ ($\Longrightarrow_\Gamma^*$). If no confusion arises, then subscript Γ can be omitted.

Definition 19 *The language generated by an* $\mathrm{FEG}_{\mathrm{PR}_{ac}}$ *system* $\Gamma = (E, A_1, \ldots, A_n, c_{init})$ *is defined by* $L(\Gamma) = \{u \mid c_{init} = (l_{1,1}, \ldots, l_{n,1}; \omega) \Longrightarrow_\Gamma^* (k_1, \ldots, k_n; u), u \in T'^*_E\}$.

The language family generated by $\mathrm{FEG}_{\mathrm{PR}_{ac}}$ systems is denoted by $\mathcal{L}(\mathrm{FEG}_{\mathrm{PR}_{ac}})$.

Let us illustrate how programmed eco–foraging systems work through an example. This example demonstrates that although the agents working separately are able to recognize finite languages only, their cooperation leads to complex behaviour.

Example 3 *Let* $L_1 = \{A'^n B'^n C'^n \mid n \geq 1\}$. *The programmed eco–foraging system* Γ *with appearance checking that generates* L, *i.e.* $L = L(\Gamma)$, *is as follows:*

$$\Gamma = (E, A_1, A_2, A_3, c_{init}),$$

such that

- $E = (V_E, T'_E, \mathcal{P}_E)$ *is the web environment, where*
  - $V_E = \{A, B, C\} \cup \{A^{(1)}, B^{(2)}, C^{(3)}\} \cup \{A', B', C'\}$,
  - $T'_E = \{A', B', C'\}$,
  - $\mathcal{P}_E = \{P_{E_1}, P_{E_2}\}$, *where*
    - $P_{E_1} = \{A \to A, B \to B, C \to C, A^{(1)} \to A', B^{(2)} \to B', C^{(3)} \to C', A' \to A', B' \to B', C' \to C'\}$,
    - $P_{E_2} = \{A \to A, B \to B, C \to C, A^{(1)} \to AA^{(1)}, B^{(2)} \to BB^{(2)}, C^{(3)} \to CC^{(3)}, A' \to A', B' \to B', C' \to C'\}$,
- $A_i = (N_i \cup N_i^{(i)}, S_i, R_i)$, $i = 1, 2, 3$, *is the i–th forager, where*
  - $N_1 = \{A\}, N_1^{(1)} = \{A^{(1)}\}, N_2 = \{B\}, N_2^{(2)} = \{B^{(2)}\}, N_3 = \{C\}, N_3^{(3)} = \{C^{(3)}\}$,
  - $S_1 = A, S_2 = B, S_3 = C$,
  - *The triplets of* R_i, $1 \leq i \leq 3$, *are of the following forms:*
    - $(l_i : X_i \to X_i^{(i)}, \sigma_i(l_i), \psi_i(l_i))$, *with* $\sigma_i(l_i) = \{l_i\}, \psi_i(l_i) = \{l_i\}$, $X_i \in N_i, X_i^{(i)} \in N_i^{(i)}$,
      - $Label(R_i) = \{l_i\}$ *and* $\sigma_i, \psi_i : Label(R_i) \to 2^{Label(R_i)}, i = 1, 2, 3$,
- $c_{init} = (l_1, l_2, l_3; \omega_{init})$, *where* $\omega_{init} = ABC$.

The derivation is as follows: $ABC \Longrightarrow_\Gamma A^{(1)}B^{(2)}C^{(3)}$, *where we have two possibilities to continue. If we apply environmental table* P_{E_1}, *then we will obtain:* $A^{(1)}B^{(2)}C^{(3)} \Longrightarrow_\Gamma A'B'C'$, *which is a terminal string. Whereas if we use* P_{E_2}, *then we will receive:* $A^{(1)}B^{(2)}C^{(3)} \Longrightarrow_\Gamma AA^{(1)}BB^{(2)}CC^{(3)}$. *Continuing the derivation with* $AA^{(1)}BB^{(2)}CC^{(3)}$, *we can employ either environmental table* P_{E_1} *or table* P_{E_2}. *Let us assume that we will use environmental table* P_{E_1} *(the continuation of the derivation for table* P_{E_2} *may be done analogously, thus it is left to the reader). As a result of the utilization of* P_{E_1}, *we will attain:* $AA^{(1)}BB^{(2)}CC^{(3)} \Longrightarrow_\Gamma A^{(1)}A'B^{(2)}B'C^{(3)}C' \Longrightarrow_\Gamma A'A'B'B'C'C'$, *which is a terminal string. We applied again environmental table* P_{E_1} *at the last step. The derivation could have been continued in a different way, if we had employed environmental table* P_{E_2} *at the last step.*

Observe that L_1 is a context–sensitive language, but it is not context–free.

4.4.2 The Power of Eco–Foraging Systems

In the sequel, we demonstrate that the class of recursively enumerable languages is exactly the same as the class of languages generated by programmed eco–foraging systems with appearance checking. It signifies that the foragers communicating only through the environment are able to identify any computable set of the environmental states. The following theorem holds:

Theorem 2 $\mathcal{L}(\mathrm{RE}) = \mathcal{L}(\mathrm{FEG}_{\mathrm{PR}_{ac}})$.

Proof
We only prove that $\mathcal{L}(\mathrm{RE}) \subseteq \mathcal{L}(\mathrm{FEG}_{\mathrm{PR}_{ac}})$, the reverse inclusion can be shown by using standard techniques. Let us assume that $L \subseteq T'^*$, $L \in \mathcal{L}(\mathrm{RE})$. Let us suppose that L is generated by $G = (N, T', S, M, \mathcal{F})$, with $M = (m_1, \ldots, m_n)$, where G is a matrix grammar in the (preliminary) 2–normal form. Let $T = \{b_j \mid 1 \leq j \leq s\}$, $T' = \{b'_j \mid b_j \in T, 1 \leq j \leq s\}$. We define the following homomorphism: $h : T' \cup N \rightarrow T \cup N$, where $h(b') = b$, for $b' \in T'$, $b \in T$ and $h(B) = B$, for $B \in N$. To prove the statement, we construct a programmed eco–foraging system Γ with appearance checking for G, such that $L(G) = L(\Gamma)$ holds. The idea is that we simulate the derivations in G by derivations in Γ.

The programmed eco–foraging system with appearance checking, able to simulate the matrix grammar is as follows:

$$\Gamma = (E, A_1, \ldots, A_n, A_{n+1}, \ldots, A_{n+s}, c_{init}),$$

where

- E is the web environment,
- $A_1, \ldots, A_n$, are the foragers that simulate the matrices of the matrix grammar G,
- $A_{n+1}, \ldots, A_{n+s}$ are special foragers that check whether the generated string is a terminal one or not according to G (in the first case, the derivation is correct, whereas in the second case, the string produced is not in the language generated by the matrix grammar), and

- c_{init} is the initial configuration.

To help the reader in following the simulation, we will denote the nonterminal letters of agents $A_{n+1}, \ldots, A_{n+s}$ by small letters. This change does not impose any restriction on the corresponding definitions.

Now we define the components of Γ. Let

$$\begin{aligned} V_E =& N \cup T \cup \{B^{(p)} \mid B \in N, 1 \leq p \leq n\} \cup \{b_j^{(n+j)}, b'_j \mid b_j \in T, 1 \leq j \leq s\} \cup \\ & \{Z_k, Z_k^{(k)} \mid 1 \leq k \leq n\} \cup \{Z_0, Z_{n+1}, Z_{n+2}\} \cup \{Z_0^{(k)}, Z_{n+1}^{(k)}, Z_{n+2}^{(k)} \mid 1 \leq k \leq n\} \cup \\ & \{C, Z_{fin}, Z_{fin}^{(fin)}, F\} \cup \{C^{i_j} \mid 1 \leq i \leq n, 1 \leq j \leq 6\}. \end{aligned}$$

The alphabet of the web environment contains all letters of the alphabet of G and other symbols that assist the simulation. These are the marker symbols $Z_0, Z_1, \ldots, Z_n, Z_{n+1}, Z_{n+2}, Z_{fin}$ and C, their indexed versions, and the trap symbol F. The trap symbol cannot be removed form the sentential form.

The idea behind the simulation is as follows: we suppose that ω' is a sentential form in G and that the corresponding word generated by Γ has the form $CZ_kZ_0Z_{n+1}Z_{n+2}\omega, 1 \leq k \leq n$, where $h(\omega') = \omega$, or its indexed version, which indicates the fact whether a given forager is active or not. The axiom is of the form $CZ_0Z_{n+1}Z_{n+2}S$, where S is the start symbol of the matrix grammar to be simulated. Both the foragers $A_i, 1 \leq i \leq n$, and the web environment can perform a rule on $Z_0, Z_1, \ldots, Z_n, Z_{n+1}, Z_{n+2}$, but only the web environment is allowed to rewrite C and Z_{fin}. Forager A_i may perform a rule only on $Z_i, 1 \leq i \leq n$, Z_0, Z_{n+1} and Z_{n+2}. Symbols $Z_0, Z_1, \ldots, Z_n, Z_{n+1}, Z_{n+2}, Z_{fin}$ and C make it possible that only the forager that simulates a matrix of G or those foragers that check whether a symbol corresponds to a letter from T', and the web environment can change the environmental word at the same time at any step of the derivation.

In the following, we present the definition of the foragers and the environmental tables and detail their roles in the simulation.

When designing the rules of A_i, we have to distinguish two cases, depending on the fact

whether m_i is with or without appearance checking, if we want to simulate the matrices of the forms $(A \to \alpha', X \to Y)$ or $(A \to \alpha', X \to \lambda)$. First, we will deal with matrices of the form $(A \to \alpha', X \to Y)$ (the second case can be treated in a similar manner, if we substitute Y with λ). Without any loss of generality, we may suppose that forager A_i simulates the work of matrix $m_i, 1 \leq i \leq n$.

Let us now consider the simulation of matrix m_i being of the form $(A \to \alpha', X \to Y)$.

First, let us suppose that m_i is without appearance checking, which means that $A \to \alpha' \notin \mathcal{F}$ and if A is not in the string, then the derivation fails. Let $A_i = (N_i \cup N_i^{(i)}, S_i, R_i)$, where $N_i = \{A, X, Z_0, Z_i, Z_{n+1}, Z_{n+2}\}$, $N_i^{(i)} = \{A^{(i)}, X^{(i)}, Z_0^{(i)}, Z_i^{(i)}, Z_{n+1}^{(i)}, Z_{n+2}^{(i)}\}$, and $S_i = Z_i$. The rule set of A_i, i.e. R_i, is as follows:

- $(l_{i_1} : Z_i \to Z_i^{(i)}, \{l_{i_2}\}, \{l_{i_1}\})$,
- $(l_{i_2} : Z_{n+2} \to Z_{n+2}^{(i)}, \{l_{i_4}\}, \{h_{i_2}\})$, $(h_{i_2} : Z_{n+2}^{(i)} \to Z_{n+2}^{(i)}, \{l_{i_6}\}, \{h_{i_2}\})$,
- $(l_{i_3} : A \to A^{(i)}, \{l_{i_5}\}, \{h_{i_2}\})$, $(h_{i_3} : A^{(i)} \to A^{(i)}, \{l_{i_5}\}, \{h_{i_2}\})$,
- $(l_{i_4} : X \to X^{(i)}, \{l_{i_3}\}, \{h_{i_2}\})$, $(h_{i_4} : X^{(i)} \to X^{(i)}, \{l_{i_3}\}, \{h_{i_2}\})$,
- $(l_{i_5} : Z_0 \to Z_0^{(i)}, \{l_{i_1}\}, \{h_{i_2}\})$, $(h_{i_5} : Z_0^{(i)} \to Z_0^{(i)}, \{l_{i_5}\}, \{h_{i_2}\})$,
- $(l_{i_6} : Z_{n+1} \to Z_{n+1}^{(i)}, \{l_{i_6}\}, \{h_{i_2}\})$, $(h_{i_6} : Z_{n+1}^{(i)} \to Z_{n+1}^{(i)}, \{l_{i_6}\}, \{h_{i_2}\})$.

We explain how the work of a matrix m_i can be simulated by the interplay of A_i and the environment. Let us assume that we have a word of the form $CZ_iZ_0Z_{n+1}Z_{n+2}\omega$ in Γ, where ω' is the corresponding sentential form in G, $h(\omega') = \omega$. The application of environmental tables $P_{E_{i_1}}, P_{E_{i_2}}, P_{E_{i_3}}, P_{E_{i_4}}, P_{E_{i_5}}, P_{E_{i_6}}$ and $P_{E_{i_7}}$ follows in succession owing to their construction. In the meantime, no other table can be employed without introducing a trap symbol. The trap symbol may also indicate the lack of the appearance checking feature.

For forager A_i that has been designated for matrix m_i, we will construct the environmental

tables. For technical reasons, we introduce the following rule sets:

$$P_E^{\{b_m,D,F\}} = \{b_m^{(n+m)} \to F, b'_m \to F, b_m \to b_m \mid b_m \in T, 1 \le m \le s\} \cup \{D \to D \mid D \in N\} \cup \{F \to F\},$$

$$P_{E_{i,1}}^{\{Z_0,Z_p\}} = \{Z_0 \to Z_0, Z_0^{(i)} \to F, Z_p \to F, Z_p^{(p)} \to F \mid 1 \le p \le n, p \ne i\},$$

$$P_{E_{i,2}}^{\{Z_0,Z_p\}} = \{Z_0 \to Z_0, Z_0^{(i)} \to Z_0^{(i)}, Z_p \to F, Z_p^{(p)} \to F \mid 1 \le p \le n, p \ne i\},$$

$$P_{E_{i,3}}^{\{Z_0,Z_p\}} = \{Z_0 \to F, Z_0^{(i)} \to Z_0, Z_p \to F, Z_p^{(p)} \to F \mid 1 \le p \le n, p \ne i\},$$

$$P_{E_{i,1}}^{\{Z_i,Z_{fin}\}} = \{Z_i \to F, Z_i^{(i)} \to Z_i^{(i)}, Z_{fin} \to F, Z_{fin}^{(fin)} \to F\},$$

$$P_{E_{i,2}}^{\{Z_i,Z_{fin}\}} = \{Z_i \to F, Z_i^{(i)} \to \lambda, Z_{fin} \to F, Z_{fin}^{(fin)} \to F\},$$

$$P_{E_{i,1}}^{\{Z_{n+1}\}} = \{Z_{n+1} \to Z_{n+1}, Z_{n+1}^{(i)} \to Z_{n+1}^{(i)}, Z_{n+1}^{(p)} \to F \mid 1 \le p \le n, p \ne i\},$$

$$P_{E_{i,2}}^{\{Z_{n+1}\}} = \{Z_{n+1} \to Z_{n+1}, Z_{n+1}^{(p)} \to F \mid 1 \le p \le n\},$$

$$P_{E_{i,1}}^{\{Z_{n+2}\}} = \{Z_{n+2} \to Z_{n+2}, Z_{n+2}^{(i)} \to Z_{n+2}^{(i)}, Z_{n+2}^{(p)} \to F \mid 1 \le p \le n, p \ne i\},$$

$$P_{E_{i,2}}^{\{Z_{n+2}\}} = \{Z_{n+2} \to Z_{n+2}, Z_{n+2}^{(i)} \to Z_{n+2}, Z_{n+2}^{(p)} \to F \mid 1 \le p \le n, p \ne i\}.$$

The first environmental table has the following form:

$$P_{E_{i_1}} = \{C \to C^{i_1}, C^{i_j} \to F, C^{k_l} \to F \mid 1 \le j \le 6, 1 \le k \le n, k \ne i, 1 \le l \le 6\} \cup \{D^{(q)} \to F \mid D \in N, 1 \le q \le n\} \cup P_E^{\{b_m,D,F\}} \cup P_{E_{i,1}}^{\{Z_0,Z_p\}} \cup P_{E_{i,1}}^{\{Z_i,Z_{fin}\}} \cup P_{E_{i,1}}^{\{Z_{n+1}\}} \cup P_{E_{i,1}}^{\{Z_{n+2}\}}.$$

After we have performed the first step of the simulation, the word will be of the form $C^{i_1} Z_i^{(i)} Z_0 Z_{n+1} Z_{n+2} \omega$, $\omega \in (N \cup T)^*$, obtained by the joint action of forager A_i ($Z_i \to Z_i^{(i)}$) and the environmental table, $P_{E_{i_1}}$. If we had employed table $P_{E_{j_1}}$ for some $1 \le j \le n$, $j \ne i$, then at the next step the trap symbol would be introduced. The trap symbol would also appear, if we had applied environmental tables $P_{E_{i_2}}, P_{E_{i_3}}, P_{E_{i_4}}$, $P_{E_{i_5}}$, $P_{E_{i_6}}$ or $P_{E_{i_7}}$.

The second environmental table is as follows:

$$P_{E_{i_2}} = \{C^{i_1} \to C^{i_2}, C^{i_j} \to F, C \to F, C^{k_l} \to F \mid 2 \le j \le 6, 1 \le k \le n, k \ne i, 1 \le l \le 6\} \cup \{D^{(q)} \to F \mid D \in N, 1 \le q \le n\} \cup P_E^{\{b_m,D,F\}} \cup P_{E_{i,1}}^{\{Z_0,Z_p\}} \cup P_{E_{i,1}}^{\{Z_i,Z_{fin}\}} \cup P_{E_{i,1}}^{\{Z_{n+1}\}} \cup P_{E_{i,1}}^{\{Z_{n+2}\}}.$$

The environmental word, which is received through the joint action of forager A_i ($Z_{n+2} \to Z_{n+2}^{(i)}$) and table $P_{E_{i_2}}$, has the form $C^{i_2} Z_i^{(i)} Z_0 Z_{n+1} Z_{n+2}^{(i)} \omega$, $\omega \in (N \cup T)^*$ after the second step of the simulation.

The third environmental table is of the form below:

$$\begin{aligned} P_{E_{i_3}} =& \{C^{i_2} \to C^{i_3}, C^{i_j} \to F \mid j = 1,3,4,5,6\} \cup \\ & \{C \to F, C^{k_l} \to F \mid 1 \le k \le n, k \ne i, 1 \le l \le 6\} \cup \\ & \{D^{(q)} \to F \mid D \in N, 1 \le q \le n\} \cup \\ & P_E^{\{b_m, D, F\}} \cup P_{E_{i,1}}^{\{Z_0, Z_p\}} \cup P_{E_{i,1}}^{\{Z_i, Z_{fin}\}} \cup P_{E_{i,1}}^{\{Z_{n+1}\}} \cup P_{E_{i,1}}^{\{Z_{n+2}\}}. \end{aligned}$$

By the action of A_i ($X \to X^{(i)}$) and $P_{E_{i_3}}$, we obtain $C^{i_3} Z_i^{(i)} Z_0 Z_{n+1} Z_{n+2}^{(i)} \gamma X^{(i)} \beta$ or $C^{i_3} Z_i^{(i)} Z_0 Z_{n+1} Z_{n+2}^{(i)} \omega$ depending on whether X is present in or absent from the environmental string. In the former case $\omega = \gamma X \beta, \beta, \gamma \in (N \cup T)^*$, in the latter ω remains unaltered.

If X has had an occurrence in the string, we have to check whether A is in the string. To this end, forager A_i employs rule $A \to A^{(i)}$. If the environmental word has not contained X, then forager A_i will apply an identical rewriting to $Z_{n+2}^{(i)}$.

The fourth environmental table has the following form:

$$\begin{aligned} P_{E_{i_4}} =& \{C^{i_3} \to C^{i_4}, C^{i_j} \to F \mid j = 1,2,4,5,6\} \cup \\ & \{C \to F, C^{k_l} \to F \mid 1 \le k \le n, k \ne i, 1 \le l \le 6\} \cup \\ & \{X^{(i)} \to X^{(i)}, X^{(r)} \to F \mid 1 \le r \le n, r \ne i\} \cup \\ & \{B^{(q)} \to F \mid B \in N \setminus \{X\}, 1 \le q \le n\} \cup \\ & P_E^{\{b_m, D, F\}} \cup P_{E_{i,1}}^{\{Z_0, Z_p\}} \cup P_{E_{i,1}}^{\{Z_i, Z_{fin}\}} \cup P_{E_{i,1}}^{\{Z_{n+1}\}} \cup P_{E_{i,1}}^{\{Z_{n+2}\}}. \end{aligned}$$

After the fourth step of the simulation, by the joint action of forager A_i and the environmental table, $P_{E_{i_4}}$, we obtain a word of one of the following forms: $C^{i_4} Z_i^{(i)} Z_0 Z_{n+1} Z_{n+2}^{(i)} \bar{\gamma} X^{(i)} \delta A^{(i)} \bar{\beta}$, if both X and A have appeared in the string, $C^{i_4} Z_i^{(i)} Z_0 Z_{n+1} Z_{n+2}^{(i)} \gamma X^{(i)} \beta$, if X has been present, but A is absent from the string, or $C^{i_4} Z_i^{(i)} Z_0 Z_{n+1} Z_{n+2}^{(i)} \omega$, if X has not occurred in it. In the first case, $\omega = \bar{\gamma} X \delta A \bar{\beta}, \bar{\beta}, \bar{\gamma}, \delta \in$

$(N \cup T)^*$, in the second case, $\omega = \gamma X \beta$, $\beta, \gamma \in (N \cup T)^*$, in the third case, ω does not change.

In the fifth step of the simulation, if both $X^{(i)}$ and $A^{(i)}$ appear in the string, forager A_i performs rule $Z_0 \to Z_0^{(i)}$, if $X^{(i)}$ is present in, but $A^{(i)}$ is absent from the string, A_i employs rule $Z_{n+2}^{(i)} \to Z_{n+2}^{(i)}$, if $X^{(i)}$ does not appear in the string, A_i substitutes Z_{n+1} for $Z_{n+1}^{(i)}$. In effect, the replacement of Z_{n+1} with $Z_{n+1}^{(i)}$ indicates that the derivation will not be successful. The web environment changes marker C^{i_4} and performs some identical rewritings in a parallel manner. Consequently, the fifth environmental table is of the form:

$$\begin{aligned} P_{E_{i_5}} =& \{C^{i_4} \to C^{i_5}, C^{i_j} \to F \mid j = 1, 2, 3, 5, 6\} \cup \\ & \{C \to F, C^{k_l} \to F \mid 1 \le k \le n, k \ne i, 1 \le l \le 6\} \cup \\ & \{A^{(i)} \to A^{(i)}, X^{(i)} \to X^{(i)}\} \cup \{A^{(r)} \to F, X^{(r)} \to F \mid 1 \le r \le n, r \ne i\} \cup \\ & \{B^{(q)} \to F \mid B \in N \setminus \{A, X\}, 1 \le q \le n\} \cup \\ & P_E^{\{b_m, D, F\}} \cup P_{E_{i,2}}^{\{Z_0, Z_p\}} \cup P_{E_{i,1}}^{\{Z_i, Z_{fin}\}} \cup P_{E_{i,1}}^{\{Z_{n+1}\}} \cup P_{E_{i,1}}^{\{Z_{n+2}\}}. \end{aligned}$$

As a result of the joint action of forager A_i and the environment, we attain one of the environmental words below: $C^{i_5} Z_i^{(i)} Z_0^{(i)} Z_{n+1} Z_{n+2}^{(i)} \bar{\gamma} X^{(i)} \delta A^{(i)} \bar{\beta}$, $C^{i_5} Z_i^{(i)} Z_0 Z_{n+1} Z_{n+2}^{(i)} \gamma X^{(i)} \beta$, or $C^{i_5} Z_i^{(i)} Z_0 Z_{n+1}^{(i)} Z_{n+2}^{(i)} \omega$.

In the sixth step of the simulation, if both $X^{(i)}$ and $A^{(i)}$ occur in the string, forager A_i rewrites again Z_i to $Z_i^{(i)}$, if $X^{(i)}$ is present, but $A^{(i)}$ is absent from the string, or if $X^{(i)}$ does not appear in the string, A_i replaces Z_{n+1} with $Z_{n+1}^{(i)}$. The sixth environmental table has the form:

$$\begin{aligned} P_{E_{i_6}} =& \{C^{i_5} \to C^{i_6}, C^{i_j} \to F \mid j = 1, 2, 3, 4, 6\} \cup \\ & \{C \to F, C^{k_l} \to F \mid 1 \le k \le n, k \ne i, 1 \le l \le 6\} \cup \\ & \{A^{(i)} \to A^{(i)}, X^{(i)} \to X^{(i)}\} \cup \{A^{(r)} \to F, X^{(r)} \to F \mid 1 \le r \le n, r \ne i\} \cup \\ & \{B^{(q)} \to F \mid B \in N \setminus \{A, X\}, 1 \le q \le n\} \cup \\ & P_E^{\{b_m, D, F\}} \cup P_{E_{i,2}}^{\{Z_0, Z_p\}} \cup P_{E_{i,2}}^{\{Z_i, Z_{fin}\}} \cup P_{E_{i,2}}^{\{Z_{n+1}\}} \cup P_{E_{i,1}}^{\{Z_{n+2}\}}. \end{aligned}$$

As a consequence, we receive one of the following environmental words: $C^{i_6} Z_0^{(i)} Z_{n+1} Z_{n+2}^{(i)} \bar{\gamma} X^{(i)} \delta A^{(i)} \bar{\beta}$, $C^{i_6} Z_0 Z_{n+1}^{(i)} Z_{n+2}^{(i)} \gamma X^{(i)} \beta$, or the derivation will not lead to a terminal string.

In the last step, forager A_i applies rule $Z_i \to Z_i^{(i)}$. In the meantime, the web environment rewrites $X^{(i)}$ to Y and $A^{(i)}$ to α, $Z_{n+1}^{(i)}$ to F and performs some other kinds of rewritings in order to make it possible for another or for the same forager to continue the work. Taking everything into consideration, the seventh environmental table has the following form:

$$\begin{aligned}
P_{E_{i_7}} =& \{C^{i_6} \to CZ_r \mid 1 \le r \le n\} \cup \{C^{i_6} \to CZ_{fin}, C^{i_j} \to F \mid 1 \le j \le 5\} \cup \\
& \{C \to F, C^{k_l} \to F \mid 1 \le k \le n, k \ne i, 1 \le l \le 6\} \cup \\
& \{A^{(i)} \to \alpha, X^{(i)} \to Y\} \cup \{A^{(r)} \to F, X^{(r)} \to F \mid 1 \le r \le n, r \ne i\} \cup \\
& \{B^{(q)} \to F \mid B \in N \setminus \{A, X\}, 1 \le q \le n\} \cup \\
& P_E^{\{b_m, D, F\}} \cup P_{E_{i,3}}^{\{Z_0, Z_p\}} \cup P_{E_{i,2}}^{\{Z_i, Z_{fin}\}} \cup P_{E_{i,2}}^{\{Z_{n+1}\}} \cup P_{E_{i,2}}^{\{Z_{n+2}\}}.
\end{aligned}$$

The environmental word obtained by the joint action of forager A_i and table $P_{E_{i_7}}$ has the form $CZ_rZ_0Z_{n+1}Z_{n+2}\bar{\gamma}Y\delta\alpha\bar{\beta}$, or $CZ_{fin}Z_0Z_{n+1}Z_{n+2}\bar{\gamma}Y\delta\alpha\bar{\beta}$, if both $X^{(i)}$ and $A^{(i)}$ appear in the string, or the derivation will not be successful.

By the construction and explanations above, the reader can easily verify that the joint work of forager A_i and the environmental tables simulates the application of m_i and only that.

Secondly, let us suppose that m_i is with appearance checking, which means that $A \to \alpha' \in \mathcal{F}$ and it can be passed over, if it cannot be applied. Let $A_i = (N_i \cup N_i^{(i)}, S_i, R_i)$, where $N_i = \{A, X, Z_0, Z_i, Z_{n+1}, Z_{n+2}\}$, $N_i^{(i)} = \{A^{(i)}, X^{(i)}, Z_0^{(i)}, Z_i^{(i)}, Z_{n+1}^{(i)}, Z_{n+2}^{(i)}\}$, and $S_i = Z_i$. The rule set R_i of A_i can be defined as follows:

- $(l_{i_1} : Z_i \to Z_i^{(i)}, \{l_{i_2}\}, \{l_{i_1}\})$,
- $(l_{i_2} : Z_{n+2} \to Z_{n+2}^{(i)}, \{l_{i_4}\}, \{h_{i_2}\})$, $(h_{i_2} : Z_{n+2}^{(i)} \to Z_{n+2}^{(i)}, \{l_{i_6}\}, \{h_{i_2}\})$,
- $(l_{i_3} : A \to A^{(i)}, \{l_{i_1}\}, \{h_{i_5}\})$, $(h_{i_3} : A^{(i)} \to A^{(i)}, \{l_{i_2}\}, \{h_{i_6}\})$,
- $(l_{i_4} : X \to X^{(i)}, \{l_{i_5}\}, \{h_{i_2}\})$, $(h_{i_4} : X^{(i)} \to X^{(i)}, \{l_{i_5}\}, \{h_{i_2}\})$,
- $(l_{i_5} : Z_0 \to Z_0^{(i)}, \{l_{i_3}\}, \{h_{i_2}\})$, $(h_{i_5} : Z_0^{(i)} \to Z_0^{(i)}, \{l_{i_1}\}, \{h_{i_2}\})$,
- $(l_{i_6} : Z_{n+1} \to Z_{n+1}^{(i)}, \{l_{i_6}\}, \{h_{i_2}\})$, $(h_{i_6} : Z_{n+1}^{(i)} \to Z_{n+1}^{(i)}, \{l_{i_6}\}, \{h_{i_2}\})$.

We present the environmental tables, which together with forager A_i, $1 \leq i \leq n$, simulate the application of m_i.

The first environmental table $P_{E_{i_1}}$ is of the following form:

$$\begin{aligned} P_{E_{i_1}} =& \{C \to C^{i_1}, C^{i_j} \to F, C^{k_l} \to F \mid 1 \leq j \leq 6, 1 \leq k \leq n, k \neq i, 1 \leq l \leq 6\} \cup \\ & \{D^{(q)} \to F \mid D \in N, 1 \leq q \leq n\} \cup \\ & P_E^{\{b_m, D, F\}} \cup P_{E_{i,1}}^{\{Z_0, Z_p\}} \cup P_{E_{i,1}}^{\{Z_i, Z_{fin}\}} \cup P_{E_{i,1}}^{\{Z_{n+1}\}} \cup P_{E_{i,1}}^{\{Z_{n+2}\}}. \end{aligned}$$

After we have performed the first step of the simulation, the word will be of the form $C^{i_1} Z_i^{(i)} Z_0 Z_{n+1} Z_{n+2} \omega$, $\omega \in (N \cup T)^*$, obtained by the joint action of forager A_i ($Z_i \to Z_i^{(i)}$) and the environmental table, $P_{E_{i_1}}$. If we had employed table $P_{E_{j_1}}$ for some $1 \leq j \leq n$, $j \neq i$, then at the next step the trap symbol would be introduced. The same is true for environmental tables $P_{E_{i_2}}, P_{E_{i_3}}, P_{E_{i_4}}, P_{E_{i_5}}$, $P_{E_{i_6}}$ and $P_{E_{i_7}}$.

The second environmental table is presented below:

$$\begin{aligned} P_{E_{i_2}} =& \{C^{i_1} \to C^{i_2}, C^{i_j} \to F, C \to F, C^{k_l} \to F \mid 2 \leq j \leq 6, 1 \leq k \leq n, k \neq i, 1 \leq l \leq 6\} \cup \\ & \{D^{(q)} \to F \mid D \in N, 1 \leq q \leq n\} \cup \\ & P_E^{\{b_m, D, F\}} \cup P_{E_{i,1}}^{\{Z_0, Z_p\}} \cup P_{E_{i,1}}^{\{Z_i, Z_{fin}\}} \cup P_{E_{i,1}}^{\{Z_{n+1}\}} \cup P_{E_{i,1}}^{\{Z_{n+2}\}}. \end{aligned}$$

The environmental word, which is received through the joint action of forager A_i ($Z_{n+2} \to Z_{n+2}^{(i)}$) and table $P_{E_{i_2}}$, has the form $C^{i_2} Z_i^{(i)} Z_0 Z_{n+1} Z_{n+2}^{(i)} \omega$, $\omega \in (N \cup T)^*$ after the second step of the simulation.

The third environmental table is of the form below:

$$\begin{aligned} P_{E_{i_3}} =& \{C^{i_2} \to C^{i_3}, C^{i_j} \to F \mid j = 1, 3, 4, 5, 6\} \cup \\ & \{C \to F, C^{k_l} \to F \mid 1 \leq k \leq n, k \neq i, 1 \leq l \leq 6\} \cup \\ & \{D^{(q)} \to F \mid D \in N, 1 \leq q \leq n\} \cup \\ & P_E^{\{b_m, D, F\}} \cup P_{E_{i,1}}^{\{Z_0, Z_p\}} \cup P_{E_{i,1}}^{\{Z_i, Z_{fin}\}} \cup P_{E_{i,1}}^{\{Z_{n+1}\}} \cup P_{E_{i,1}}^{\{Z_{n+2}\}}. \end{aligned}$$

By the action of A_i ($X \to X^{(i)}$) and $P_{E_{i_3}}$, we obtain $C^{i_3} Z_i^{(i)} Z_0 Z_{n+1} Z_{n+2}^{(i)} \gamma X^{(i)} \beta$, or $C^{i_3} Z_i^{(i)} Z_0 Z_{n+1} Z_{n+2}^{(i)} \omega$, depending on whether X is present in or absent from the environ-

mental string. In the former case $\omega = \gamma X \beta, \beta, \gamma \in (N \cup T)^*$, in the latter ω remains unaltered.

In the fourth step, depending on the occurrence of $X^{(i)}$ in the environmental word, forager A_i either replaces Z_0 with $Z_0^{(i)}$, or applies an identical rewriting to $Z_{n+2}^{(i)}$. The fourth environmental table has to be constructed as follows:

$$\begin{aligned} P_{E_{i_4}} =& \{C^{i_3} \to C^{i_4}, C^{i_j} \to F \mid j = 1,2,4,5,6\} \cup \\ & \{C \to F, C^{k_l} \to F \mid 1 \leq k \leq n, k \neq i, 1 \leq l \leq 6\} \cup \\ & \{X^{(i)} \to X^{(i)}, X^{(r)} \to F \mid 1 \leq r \leq n, r \neq i\} \cup \\ & \{B^{(q)} \to F \mid B \in N \setminus \{X\}, 1 \leq q \leq n\} \cup \\ & P_E^{\{b_m, D, F\}} \cup P_{E_{i,1}}^{\{Z_0, Z_p\}} \cup P_{E_{i,1}}^{\{Z_i, Z_{fin}\}} \cup P_{E_{i,1}}^{\{Z_{n+1}\}} \cup P_{E_{i,1}}^{\{Z_{n+2}\}}. \end{aligned}$$

After the fourth step of the simulation we obtain by the joint action of forager A_i and the environmental table, $P_{E_{i_4}}$, a word of one of the following forms: $C^{i_4} Z_i^{(i)} Z_0^{(i)} Z_{n+1} Z_{n+2}^{(i)} \gamma X^{(i)} \beta$, if $X^{(i)}$ occurs in the string, or $C^{i_4} Z_i^{(i)} Z_0 Z_{n+1} Z_{n+2}^{(i)} \omega$ otherwise. In the first case, $\omega = \gamma X \beta$, $\beta, \gamma \in (N \cup T)^*$, in the second case, ω does not change.

In the fifth step of the simulation, we control the presence of A in the string or introduce $Z_{n+1}^{(i)}$ into the string. In the first case, agent A_i employs rule $A \to A^{(i)}$, in the second case, it substitutes Z_{n+1} for $Z_{n+1}^{(i)}$. The fifth environmental table is of the form:

$$\begin{aligned} P_{E_{i_5}} =& \{C^{i_4} \to C^{i_5}, C^{i_j} \to F \mid j = 1,2,3,5,6\} \cup \\ & \{C \to F, C^{k_l} \to F \mid 1 \leq k \leq n, k \neq i, 1 \leq l \leq 6\} \cup \\ & \{A^{(i)} \to A^{(i)}, X^{(i)} \to X^{(i)}\} \cup \{A^{(r)} \to F, X^{(r)} \to F \mid 1 \leq r \leq n, r \neq i\} \cup \\ & \{B^{(q)} \to F \mid B \in N \setminus \{A, X\}, 1 \leq q \leq n\} \cup \\ & P_E^{\{b_m, D, F\}} \cup P_{E_{i,2}}^{\{Z_0, Z_p\}} \cup P_{E_{i,1}}^{\{Z_i, Z_{fin}\}} \cup P_{E_{i,1}}^{\{Z_{n+1}\}} \cup P_{E_{i,1}}^{\{Z_{n+2}\}}. \end{aligned}$$

As a result of the joint action of forager A_i and the environment, we attain one of the environmental words $C^{i_5} Z_i^{(i)} Z_0^{(i)} Z_{n+1} Z_{n+2}^{(i)} \gamma' X^{(i)} \delta A^{(i)} \beta'$, $C^{i_5} Z_i^{(i)} Z_0^{(i)} Z_{n+1} Z_{n+2}^{(i)} \gamma X^{(i)} \beta$, or $C^{i_5} Z_i^{(i)} Z_0 Z_{n+1}^{(i)} Z_{n+2}^{(i)} \omega$. In the first case, $\omega = \bar{\gamma} X \delta A \bar{\beta}, \bar{\beta}, \bar{\gamma}, \delta \in (N \cup T)^*$, in the second case, $\omega = \gamma X \beta$, $\beta, \gamma \in (N \cup T)^*$, in the third case, ω does not change.

In the sixth step, if $X^{(i)}$ and $A^{(i)}$ are in the string, then forager A_i attempts to employ $Z_i \to Z_i^{(i)}$. If only $X^{(i)}$ occurs in the environmental word, then forager A_i rewrites $Z_0^{(i)}$ identically. Should $X^{(i)}$ be absent from the string, then A_i tries to substitute Z_{n+1} for $Z_{n+1}^{(i)}$. The sixth environmental table can be constructed as follows:

$$\begin{aligned}
P_{E_{i_6}} =&\{C^{i_5} \to C^{i_6}, C^{i_j} \to F \mid j = 1,2,3,4,6\} \cup \\
&\{C \to F, C^{k_l} \to F \mid 1 \le k \le n, k \ne i, 1 \le l \le 6\} \cup \\
&\{A^{(i)} \to A^{(i)}, X^{(i)} \to X^{(i)}\} \cup \{A^{(r)} \to F, X^{(r)} \to F \mid 1 \le r \le n, r \ne i\} \cup \\
&\{B^{(q)} \to F \mid B \in N \setminus \{A, X\}, 1 \le q \le n\} \cup \\
&P_E^{\{b_m, D, F\}} \cup P_{E_{i,2}}^{\{Z_0, Z_p\}} \cup P_{E_{i,1}}^{\{Z_i, Z_{fin}\}} \cup P_{E_{i,1}}^{\{Z_{n+1}\}} \cup P_{E_{i,1}}^{\{Z_{n+2}\}}.
\end{aligned}$$

Through the joint action of forager A_i and the environment we receive after the sixth step $C^{i_6} Z_i^{(i)} Z_0^{(i)} Z_{n+1} Z_{n+2}^{(i)} \bar{\gamma} X^{(i)} \delta A^{(i)} \bar{\beta}$, $C^{i_6} Z_i^{(i)} Z_0^{(i)} Z_{n+1} Z_{n+2}^{(i)} \gamma X^{(i)} \beta$, or $C^{i_6} Z_i^{(i)} Z_0 Z_{n+1}^{(i)} Z_{n+2}^{(i)} \omega$. In the first case, $\omega = \bar{\gamma} X \delta A \bar{\beta}, \bar{\beta}, \bar{\gamma}, \delta \in (N \cup T)^*$, in the second case, $\omega = \gamma X \beta, \beta, \gamma \in (N \cup T)^*$, in the third case, ω does not change.

In the last step, forager A_i applies rule $Z_i \to Z_i^{(i)}$, if $X^{(i)}$ is in the string. If $X^{(i)}$ does not have an occurrence in the string, agent A_i tries to rewrite Z_{n+2} to $Z_{n+2}^{(i)}$. In the meantime, the web environment replaces $X^{(i)}$ with Y, $A^{(i)}$ with α (if $A^{(i)}$ is not present in the string, then rule $A^{(i)} \to \alpha$ will not be employed), and $Z_{n+1}^{(i)}$ with F and performs some other kinds of rewritings in order to make it possible for another or for the same forager to continue the work. Taking everything into consideration, the seventh environmental table must have the following form:

$$\begin{aligned}
P_{E_{i_7}} =&\{C^{i_6} \to C Z_r \mid 1 \le r \le n\} \cup \{C^{i_6} \to C Z_{fin}, C^{i_j} \to F \mid 1 \le j \le 5\} \cup \\
&\{C \to F, C^{k_l} \to F \mid 1 \le k \le n, k \ne i, 1 \le l \le 6\} \cup \\
&\{A^{(i)} \to \alpha, X^{(i)} \to Y\} \cup \{A^{(r)} \to F, X^{(r)} \to F \mid 1 \le r \le n, r \ne i\} \cup \\
&\{B^{(q)} \to F \mid B \in N \setminus \{A, X\}, 1 \le q \le n\} \cup \\
&P_E^{\{b_m, D, F\}} \cup P_{E_{i,3}}^{\{Z_0, Z_p\}} \cup P_{E_{i,2}}^{\{Z_i, Z_{fin}\}} \cup P_{E_{i,2}}^{\{Z_{n+1}\}} \cup P_{E_{i,2}}^{\{Z_{n+2}\}}.
\end{aligned}$$

The environmental word obtained by the joint action of forager A_i and table $P_{E_{i_7}}$ has the form $C Z_r Z_0 Z_{n+1} Z_{n+2} \bar{\gamma} Y \delta \alpha \bar{\beta}$ ($C Z_{fin} Z_0 Z_{n+1} Z_{n+2} \bar{\gamma} Y \delta \alpha \bar{\beta}$), $C Z_r Z_0 Z_{n+1} Z_{n+2} \gamma Y \beta$ ($C Z_{fin} Z_0 Z_{n+1} Z_{n+2} \gamma Y \beta$), or the derivation will not lead to a terminal word.

As in the previous case, the reader can verify that the joint work of forager A_i and the environmental tables simulates the application of m_i and only that.

At some stage of the derivation, we guess whether ω is a terminal word or not (the environmental word is of the form $CZ_{fin}Z_0Z_{n+1}Z_{n+2}\omega$). We check this conjecture by the joint action of foragers A_{n+j}, $1 \leq j \leq s$, and environmental table $P_{E_{fin}}$.

Let $A_{n+j} = (N_{n+j} \cup N_{n+j}^{(n+j)}, S_{n+j}, R_{n+j})$, where $N_{n+j} = \{b_j\}$, $N_{n+j}^{(n+j)} = \{b_j^{(n+j)}\}$, and $S_{n+j} = b_j$, $1 \leq j \leq s$. The rule set of A_{n+j}, i.e. R_{n+j}, $1 \leq j \leq s$, is defined as follows ($b_j \in T = \{b_j \mid 1 \leq j \leq s\}$): $(l_j : b_j \rightarrow b_j^{(n+j)}, \{l_j\}, \{l_j\})$. Observe that $(l_j : b_j \rightarrow b_j^{(n+j)}, \{l_j\}, \{l_j\})$ is the only rule of forager A_{n+j}, thus its initial rule, as well.

Let

$$\begin{aligned}
P_{E_{fin}} =& \{C^{i_p} \rightarrow F \mid 1 \leq i \leq n, 1 \leq p \leq 6\} \cup \{C \rightarrow \lambda\} \cup \\
& \{D \rightarrow F, D^{(q)} \rightarrow F \mid D \in N, 1 \leq q \leq n\} \cup \\
& \{b_j^{(n+j)} \rightarrow b'_j, b'_j \rightarrow b'_j, b_j \rightarrow b_j \mid b_j \in T, 1 \leq j \leq s\} \cup \{F \rightarrow F\} \cup \\
& \{Z_0 \rightarrow \lambda, Z_0^{(k)} \rightarrow F, Z_{fin} \rightarrow \lambda, Z_{fin}^{(fin)} \rightarrow F \mid 1 \leq k \leq n\} \cup \\
& \{Z_k \rightarrow \lambda, Z_k^{(k)} \rightarrow F \mid 1 \leq k \leq n\} \cup \{Z_{n+1} \rightarrow \lambda, Z_{n+1}^{(k)} \rightarrow F \mid 1 \leq k \leq n\} \cup \\
& \{Z_{n+2} \rightarrow \lambda, Z_{n+2}^{(k)} \rightarrow F \mid 1 \leq k \leq n\}.
\end{aligned}$$

Foragers A_{n+j}, $1 \leq j \leq s$, rewrite terminal letters $b_j, 1 \leq j \leq s$, to their indexed versions $b_j^{(n+j)}$, and in the meantime environmental table $P_{E_{fin}}$ deletes the marker symbols, introduces F for the letters from N and rewrites letters $b_j^{(n+j)}$, $1 \leq j \leq s$, to b'_j. The work of foragers $A_{n+j}, 1 \leq j \leq s$, cannot be interfered with the work of the other foragers. The joint work of foragers A_{n+j}, $1 \leq j \leq s$, and environmental table $P_{E_{fin}}$ is iterated as many times as it is necessary. At the end of the procedure, if the word obtained is a string over T', then a word that can be generated by G is received.

It only remains to be shown how the simulation begins. The initial state of the web environment is $\omega = CZ_0Z_{n+1}Z_{n+2}S$. We have to simulate matrix $(S \rightarrow AX)$. Let us

assume that the web environment has a table $P_{E_{start}}$ that performs this simulation, where

$$\begin{aligned}P_{E_{start}} =&\{C \to CZ_k, Z_k \to F, Z_k^{(k)} \to F \mid 1 \leq k \leq n\} \cup \{C^{p_q} \to F \mid 1 \leq p \leq n, 1 \leq q \leq 6\} \cup \\ &\{S \to AX\} \cup \{D \to F \mid D \in N \setminus \{S\}\} \cup \{B^{(m)} \to F \mid B \in N, 1 \leq m \leq n\} \cup \\ &\{b_j \to F, b_j' \to F, b_j^{(n+j)} \to F \mid b_j \in T, 1 \leq j \leq s\} \cup \\ &\{Z_0 \to Z_0, Z_0^{(k)} \to F, Z_{fin} \to F, Z_{fin}^{(fin)} \to F \mid 1 \leq k \leq n\} \cup \\ &\{Z_{n+1} \to Z_{n+1}, Z_{n+1}^{(k)} \to F \mid 1 \leq k \leq n\} \cup \\ &\{Z_{n+2} \to Z_{n+2}, Z_{n+2}^{(k)} \to F \mid 1 \leq k \leq n\} \cup \{F \to F\}.\end{aligned}$$

Initially, all the foragers are inactive, since there is not a $Z_k, 1 \leq k \leq n$, in the sentential form. The derivation is as follows: $CZ_0Z_{n+1}Z_{n+2}S \Longrightarrow_\Gamma CZ_kZ_0Z_{n+1}Z_{n+2}AX$ for some $k, 1 \leq k \leq n$. Since S does not occur in any other matrix, this is the only way how the simulation can begin.

Owing to the form of the tables, it can be seen that all but no more words than the elements of $L(G)$ can be derived. Hence the theorem is verified. ∎

4.4.3 Eco–Foraging Systems with Time

Secondly, we move on to eco–foraging systems with web pages having lifetime. We modify the alphabet of agents to keep track of the aging of the web environment [78]. If the lifetime of a web page is 0, it signifies that the web page is no longer recognizable by any of the foragers.

Definition 20 *The web environment with n foragers with time, $n \geq 1$, is a construction*

$$E = (V_E, T_E, \mathcal{P}_E),$$

such that

- V_E *is a finite alphabet, where* $V_E = V_M \cup T_E \cup V_N \cup \bar{V}_N$ *with* $V_N = \bigcup_{i=1}^{n} N_i$ *and* $\bar{V}_N = \bigcup_{i=1}^{n} N_i^{(i)}$, $n \geq 1$, *where*

- V_M *is a finite set,*
- $N_i = \bigcup_{j=1}^{s_i} N_{i,j}$, $N_{i,j} = \bigcup_{k=0}^{t_{i,j}} N_{i,j}(k) = \bigcup_{k=0}^{t_{i,j}} \{X_{i,j}(k)\}$,
- $N_i^{(i)} = \bigcup_{j=1}^{s_i} N_{i,j}^{(i)}$, $N_{i,j}^{(i)} = \bigcup_{k=0}^{t_{i,j}} N_{i,j}^{(i)}(k) = \bigcup_{k=0}^{t_{i,j}} \{X_{i,j}^{(i)}(k)\}$,
- T_E *is a finite alphabet, and*
- V_M, T_E, V_N *and* $\bar{V}_N$ *are pairwise disjoint sets,*

- $\mathcal{P}_E = \{P_{E_1}, \ldots, P_{E_y}\}$, *where* P_{E_q}, $1 \leq q \leq y$, *is a finite set of rules of the following forms,* $V_{N_{max}} = \bigcup_{p=1}^{n} \bigcup_{r=1}^{s_p} N_{p,r}(t_{p,r})$, $t_{p,r} \geq 1$, $1 \leq i \leq n, 1 \leq j \leq s_i$, $1 \leq k \leq t_{i,j}$*:*
  - $X_{i,j}(k) \to X_{i,j}(k-1)$, *where* $X_{i,j}(k-1), X_{i,j}(k) \in N_{i,j}$,
  - $Y \to \alpha$, *where* $Y \in \bigcup_{i=1}^{n} N_i$, *and* $\alpha \in (T_E \cup V_{N_{max}})^*$,
  - $X_{i,j}^{(i)}(k) \to \beta$, *where* $X_{i,j}^{(i)}(k) \in N_{i,j}^{(i)}$, *and* $\beta \in (T_E \cup V_{N_{max}})^* \cup (T_E \cup V_{N_{max}})^* X_{i,j}^{(i)}(k-1)(T_E \cup V_{N_{max}})^*$, $X_{i,j}^{(i)}(k-1) \in N_{i,j}^{(i)}$,
  - $X_{i,j}(0) \to X_{i,j}(0)$, $X_{i,j}(0) \in N_{i,j}$,
  - $X_{i,j}^{(i)}(0) \to X_{i,j}^{(i)}(0)$, $X_{i,j}^{(i)}(0) \in N_{i,j}^{(i)}$, *or*
  - $U \to \gamma$, *where* $U \in V_M$ *and* $\gamma \in (T_E \cup V_{N_{max}})^* \cup (T_E \cup V_{N_{max}})^* V_M (T_E \cup V_{N_{max}})^*$.

  Moreover, any rule set in $\mathcal{P}_E$ *is complete, i.e. for any* $c \in V_E$*, there is at least one rule in any* P_{E_q}, $1 \leq q \leq y$.

If the lifetime of the web pages is included, then the interpretation of the various components of the web environment is analogous to the one presented for Def. 15. Therefore herein we emphasize only the differences. The alphabet of an agent also contains the information about the lifetime of the web pages that the agent is able to recognize. We assign a maximal lifetime to each web page. If the environment rewrites a web page and the web page will still be present in the environmental string, then the lifetime of the web page will be reduced by one regardless of whether any agents have managed to identify the web page or not. The lifetime of the newly introduced web pages will be maximal (in $V_{N_{max}}$). We do not assign lifetime to the elements of V_M and T_E. Furthermore, T_E is disjoint from V_M, V_N and $\bar{V}_N$. While in Def. 15 the elements of T_E can be rewritten by the agents, in this definition they can be changed by the environment only.

Definition 21 *A programmed eco–foraging system with appearance checking with time (an* $\mathrm{FEG}^{time}_{\mathrm{PR}_{ac}}$ *system) of degree* n, $n \geq 1$, *is a construction*

$$\Gamma = (E, A_1, \ldots, A_n, c_{init}),$$

such that

- $E = (V_E, T_E, \mathcal{P}_E)$ *is the web environment with time (see Def.20),*
- $A_i = (\bar{N}_i \cup N_i^{(i)}, S_i, R_i)$, $1 \leq i \leq n$, *is the* i*–th forager, a programmed grammar scheme with appearance checking, where*
  - $\bar{N}_i \cup N_i^{(i)}$ *is the nonterminal alphabet of the* i*–th forager,* $\bar{N}_i = N_i \setminus N_i(0)$, *where* $N_i(0) = \bigcup_{j=1}^{s_i} N_{i,j}(0) = \bigcup_{j=1}^{s_i} \{X_{i,j}(0)\}$ *(see Def.20),*
  - $S_i \in \bar{N}_i$ *is the start symbol of the* i*–th forager,* $S_i = X_{i,1}$,
  - R_i *is a finite set of rules of the following forms:*
    - $(l_{i,1}(k) : S_i(k) \rightarrow S_i^{(i)}(k-1), \sigma_i(l_{i,1}(k)), \psi_i(l_{i,1}(k)))$, *with* $\sigma_i(l_{i,1}(k)) = \bigcup_{q=1}^{s_p} l_{p,q}$, $\{l_{p,1}, \ldots, l_{p,s_p}\} \subseteq \{l_{i,1}, \ldots, l_{i,s_i}\}$, $\psi_i(l_{i,1}(k)) = l_{i,1}$, $1 \leq k \leq t_{i,1}$, *is called the initial rule of the* i*–th forager, where* $l_{i,j} = \{l_{i,j}(z) \mid 1 \leq z \leq t_{i,j}\}$, $1 \leq j \leq s_i$,
    - $(l_{i,j}(k) : X_{i,j}(k) \rightarrow X_{i,j}^{(i)}(k-1), \sigma_i(l_{i,j}(k)), \psi_i(l_{i,j}(k)))$, $X_{i,j}(k) \in \bar{N}_i \setminus \{S_i\}, X_{i,j}^{(i)}(k-1) \in N_i^{(i)} \setminus \{S_i^{(i)}\}$, $2 \leq j \leq s_i$, *with* $\sigma_i(l_{i,j}(k)) = \bigcup_{q=1}^{s_p} l_{p,q}$, $\{l_{p,1}, \ldots, l_{p,s_p}\} \subseteq \{l_{i,1}, \ldots, l_{i,s_i}\}$, $\psi_i(l_{i,j}(k)) = \bigcup_{q'=2}^{s_p} h_{p,q'}$, $\{h_{p,2}, \ldots, h_{p,s_p}\} \subseteq \{h_{i,2}, \ldots, h_{i,s_i}\}$, *where* $l_{i,j} = \{l_{i,j}(z) \mid 1 \leq z \leq t_{i,j}\}$, $1 \leq j \leq s_i$, $h_{i,j'} = \{h_{i,j'}(z) \mid 1 \leq z \leq t_{i,j'}\}$, $2 \leq j' \leq s_i$, *or*
    - $(h_{i,j}(k) : X_{i,j}^{(i)}(k) \rightarrow X_{i,j}^{(i)}(k-1), \sigma_i(h_{i,j}(k)), \psi_i(h_{i,j}(k)))$, $X_{i,j}^{(i)}(k), X_{i,j}^{(i)}(k-1) \in N_i^{(i)} \setminus \{S_i^{(i)}\}$, $2 \leq j \leq s_i$, *with* $\sigma_i(h_{i,j}(k)) = \bigcup_{q=1}^{s_p} l_{p,q}$, $\{l_{p,1}, \ldots, l_{p,s_p}\} \subseteq \{l_{i,1}, \ldots, l_{i,s_i}\}$, $\psi_i(h_{i,j}(k)) = \bigcup_{q'=2}^{s_p} h_{p,q'}$, $\{h_{p,2}, \ldots, h_{p,s_p}\} \subseteq \{h_{i,2}, \ldots, h_{i,s_i}\}$, *where* $l_{i,j} = \{l_{i,j}(z) \mid 1 \leq z \leq t_{i,j}\}$, $1 \leq j \leq s_i$, $h_{i,j'} = \{h_{i,j'}(z) \mid 1 \leq z \leq t_{i,j'}\}$, $2 \leq j' \leq s_i$, *and*
      - $Label(R_i) = l_{i,1} \cup \ldots \cup l_{i,s_i} \cup h_{i,2} \cup \ldots \cup h_{i,s_i}$ *is the set of labels of* R_i.

- $c_{init} = (l_{1,1}(t_{1,1}), \ldots, l_{n,1}(t_{n,1}); \omega_{init})$ *is called the initial configuration of* Γ*, where* $l_{i,1}(t_{i,1})$*,* $t_{i,1} \geq 1$*, is the label of the initial rule of the* i*–th forager with the corresponding maximal time,* $1 \leq i \leq n$*,* $\omega_{init} = z_1 X_{j_1}(t_{j_1}) z_2 \ldots z_k X_{j_k}(t_{j_k}) z_{k+1}$*,* $X_{j_h}(t_{j_h}) \in \bar{N}_{j_h}$*,* $t_{j_h} \geq 1$*,* $z_1 \in T_E^* \cup T_E^* V_M T_E^*$*,* $z_l \in T_E^*, 1 \leq h \leq k, 1 \leq l \leq k+1$*, and for some* k*,* $0 \leq k \leq n$*,* $\{j_1, \ldots, j_k\} \subseteq \{1, \ldots, n\}$*. The string* ω_{init} *is called the initial state of the web environment of* Γ *or the initial environmental state.*

In Def. 21 when the agent tries to visit a not yet discovered web page employing rules of the form $X_{i,j}(k) \rightarrow X_{i,j}^{(i)}(k-1)$, $X_{i,j}(k) \in \bar{N}_i, X_{i,j}^{(i)}(k-1) \in N_i^{(i)}$, $1 \leq i \leq n, 1 \leq j \leq s_i$, $1 \leq k \leq t_{i,j}$, then the lifetime of the web page will be reduced by one, if the application of the rule has been successful. Should the agent go to a web page that it has discovered previously using rules of the form $X_{i,j}^{(i)}(k) \rightarrow X_{i,j}^{(i)}(k-1)$, $X_{i,j}^{(i)}(k-1), X_{i,j}^{(i)}(k) \in N_i^{(i)}$, $1 \leq i \leq n, 2 \leq j \leq s_i$, $1 \leq k \leq t_{i,j}$, then the lifetime of the corresponding web page will be decreased by one again. In the axiom, the lifetime of the nonterminal letters of those agents that are able to commence their work is maximal. Notice that ω_{init} contains at most one symbol from V_M.

In the sequel, we define the way in which eco–foraging systems with time work.

Definition 22 *Let* $\Gamma = (E, A_1, \ldots, A_n, c_{init})$*, be an* $\mathrm{FEG}_{\mathrm{PR}_{ac}}^{time}$ *system of degree* n*,* $n \geq 1$ *(see Def. 21). An* $(n+1)$*–tuple* $c = (q_{1,j_1}(k_{1,j_1}), \ldots, q_{n,j_n}(k_{n,j_n}); \omega_E)$*, where* $q_{i,j_i}(k_{i,j_i}) \in Label(R_i)$*,* $1 \leq k_{i,j_i} \leq t_{i,j_i}$*,* $1 \leq i \leq n$*,* $1 \leq j_i \leq s_i$*, and* $\omega_E \in V_E^*$*, is called a configuration of* Γ*.* ω_E *is the state of the web environment of* Γ *in configuration* c *or the environmental state in configuration* c*.*

Definition 23 *Let* $\Gamma = (E, A_1, \ldots, A_n, c_{init})$*,* $1 \leq t_{i,1}$*,* $1 \leq i \leq n$*, be an* $\mathrm{FEG}_{\mathrm{PR}_{ac}}^{time}$ *system of degree* n *(see Def. 21). Let* $c_1 = (q_{1,j_1}(k_{1,j_1}), \ldots, q_{n,j_n}(k_{n,j_n}); \omega_E)$*,* $c_2 = (q'_{1,j_1}(k'_{1,j_1}), \ldots, q'_{n,j_n}(k'_{n,j_n}); \omega'_E)$ *be two configurations of* Γ*,* $1 \leq k_{i,j_i}, k'_{i,j_i} \leq t_{i,j_i}$*,* $1 \leq i \leq n$*,* $1 \leq j_i \leq s_i$*, and* $\omega_E, \omega'_E \in V_E^*$*. We say that* c_1 *directly derives* c_2*, written as* $c_1 \Longrightarrow_\Gamma c_2$*, if the following conditions hold:*

1. $\omega_E = u_1 \alpha_{i_1}(k_{i_1}) u_2 \ldots u_r \alpha_{i_r}(k_{i_r}) u_{r+1}$ *and* $\omega''_E = u_1 \beta_{i_1}(k_{i_1} - 1) u_2 \ldots u_r \beta_{i_r}(k_{i_r} - 1) u_{r+1}$*, where for some* r*,* $0 \leq r \leq n$*,* $\{i_1, \ldots, i_r\} \subseteq \{1, \ldots, n\}$*,* $\alpha_{i_j}(k_{i_j}) \in \bar{N}_j \cup N_j^{(j)}, \beta_{i_j}(k_{i_j} - 1) \in N_j^{(j)}$*,* $1 \leq k_{i_j} \leq t_{i_j}$*,* $1 \leq j \leq r$*,* $u_h \in V_E^*$*,* $1 \leq h \leq r+1$*,*

2. $(q_{i_j}(k_{i_j}) : \alpha_{i_j}(k_{i_j}) \rightarrow \beta_{i_j}(k_{i_j} - 1), \sigma(q_{i_j}(k_{i_j})), \psi(q_{i_j}(k_{i_j}))) \in R_{i_j}$ *and* $q'_{i_j}(k'_{i_j}) \in \sigma(q_{i_j}(k_{i_j}))$, $1 \leq j \leq r$,

3. *there is no* $m \in \{1, \ldots, n\} \setminus \{i_1, \ldots, i_r\}$, *such that* $(q_m(k_m) : \alpha_m(k_m) \rightarrow \beta_m(k_m - 1), \sigma(q_m(k_m)), \psi(q_m(k_m))) \in R_m$ *can be applied to* $u_1 u_2 \ldots u_{r+1}$, $1 \leq k_m \leq t_m$,

4. $q'_m(k'_m) \in \psi(q_m(k_m))$ *for* $m \in \{1, \ldots, n\} \setminus \{i_1, \ldots, i_r\}$, $1 \leq k'_m \leq t'_m$,

5. $\omega'_E = v_1 \beta_{i_1}(k_{i_1} - 1) v_2 \ldots v_r \beta_{i_r}(k_{i_r} - 1) v_{r+1}$, *where* $u_1 \ldots u_{r+1} \Longrightarrow v_1 \ldots v_{r+1}$ *is a* 0L *rewriting according to some* P_{E_q}, $1 \leq q \leq y$, *where* $P_{E_q} \in \mathcal{P}_E$.

The function of a programmed eco–foraging system with time is analogous to the function of a programmed eco–foraging system without time (see Def. 18), therefore herein we present only the differences. In Def. 23, the agents that participate in the derivation will reduce the lifetime of the web pages they visit. The lifetime of the web pages remaining still present in the environment or being introduced at the given step, will be modified by the environment as it is described in the remark following Def. 20.

The transitive (and reflexive) closure of $\Longrightarrow_\Gamma$ is denoted by $\Longrightarrow_\Gamma^+$ ($\Longrightarrow_\Gamma^*$). If no confusion arises, then subscript Γ can be omitted.

Definition 24 *The language generated by an* $\mathrm{FEG}_{\mathrm{PR}_{ac}}^{time}$ *system* $\Gamma = (E, A_1, \ldots, A_n, c_{init})$ *is defined by* $L(\Gamma) = \{y \mid c_{init} = (l_{1,1}(t_{1,1}), \ldots, l_{n,1}(t_{n,1}); \omega_{init}) \Longrightarrow_\Gamma^* (q_{1,j_1}(k_{1,j_1}), \ldots, q_{n,j_n}(k_{n,j_n}); y), y \in T_E^*\}$.

The family of languages generated by $\mathrm{FEG}_{\mathrm{PR}_{ac}}^{time}$ systems is denoted by $\mathcal{L}(\mathrm{FEG}_{\mathrm{PR}_{ac}}^{time})$.

Let us now illustrate how programmed eco–foraging systems with time work through an example.

Example 4 *Let* $L_2 = \{A^n B^n C^n \mid n \geq 1\}$. *The programmed eco–foraging system* Γ *with time with appearance checking that generates* L_2, *i.e.* $L_2 = L(\Gamma)$, *is as follows:*

$$\Gamma = (E, A_1, A_2, A_3, c_{init}),$$

such that

- $E = (V_E, T_E, \mathcal{P}_E)$ *is the web environment, where*
  - $V_E = \{A(t), B(t), C(t) \mid 0 \leq t \leq 3\} \cup \{A^{(1)}(t), B^{(2)}(t), C^{(3)}(t) \mid 0 \leq t \leq 3\} \cup \{A, B, C\}$,
  - $T_E = \{A, B, C\}$,
  - $\mathcal{P}_E = \{P_{E_1}, P_{E_2}\}$, *where*
    - $P_{E_1} = \{A(0) \rightarrow A(0), A(t) \rightarrow A(t-1), B(0) \rightarrow B(0), B(t) \rightarrow B(t-1), C(0) \rightarrow C(0), C(t) \rightarrow C(t-1), A^{(1)}(0) \rightarrow A^{(1)}(0), A^{(1)}(t) \rightarrow A, B^{(2)}(0) \rightarrow B^{(2)}(0), B^{(2)}(t) \rightarrow B, C^{(3)}(0) \rightarrow C^{(3)}(0), C^{(3)}(t) \rightarrow C, A \rightarrow A, B \rightarrow B, C \rightarrow C \mid 1 \leq t \leq 3\}$,
    - $P_{E_2} = \{A(0) \rightarrow A(0), A(t) \rightarrow A(t-1), B(0) \rightarrow B(0), B(t) \rightarrow B(t-1), C(0) \rightarrow C(0), C(t) \rightarrow C(t-1), A^{(1)}(0) \rightarrow A^{(1)}(0), A^{(1)}(t) \rightarrow A(3)A^{(1)}(t-1), B^{(2)}(0) \rightarrow B^{(2)}(0), B^{(2)}(t) \rightarrow B(3)B^{(1)}(t-1), C^{(3)}(0) \rightarrow C^{(3)}(0), C^{(3)}(t) \rightarrow C(3)C^{(1)}(t-1), A \rightarrow A, B \rightarrow B, C \rightarrow C \mid 1 \leq t \leq 3\}$,
- $A_i = (\bar{N}_i \cup N_i^{(i)}, S_i, R_i)$, $i = 1, 2, 3$, *is the* i*–th forager, where*
  - $\bar{N}_1 = \{A(t) \mid 1 \leq t \leq 3\}, N_1^{(1)} = \{A^{(1)}(t) \mid 0 \leq t \leq 3\}, \bar{N}_2 = \{B(t) \mid 1 \leq t \leq 3\}, N_2^{(2)} = \{B^{(2)}(t) \mid 0 \leq t \leq 3\}, \bar{N}_3 = \{C(t) \mid 1 \leq t \leq 3\}, N_3^{(3)} = \{C^{(3)}(t) \mid 0 \leq t \leq 3\}$,
  - *The triplets of* R_i *are of the following forms:*
    - $(l_i(t) : X_i(t) \rightarrow X_i^{(i)}(t-1), \sigma_i(l_i(t)), \psi_i(l_i(t)))$, *with* $\sigma_i(l_i(t)) = l_i, \psi_i(l_i(t)) = l_i$, $l_i = \{l_i(t) \mid 1 \leq t \leq 3\}$, $X_i(t) \in \bar{N}_i$, $X_i^{(i)}(t-1) \in N_i^{(i)}$, $1 \leq t \leq 3$, *where*
      - $Label(R_i) = \{l_i(t) \mid 1 \leq t \leq 3\}$ *and* $\sigma_i, \psi_i : Label(R_i) \rightarrow 2^{Label(R_i)}, i = 1, 2, 3$,
- $c_{init} = (l_1(3), l_2(3), l_3(3); \omega_{init})$, *where* $\omega_{init} = A(3)B(3)C(3)$.

At the first step all foragers are active. They rewrite different symbols from the environmental string to their indexed versions and the lifetime of the web pages will be reduced. Forager A_1 *changes* $A(3)$ *to* $A^{(1)}(2)$, *forager* A_2 $B(3)$ *to* $B^{(2)}(2)$ *and forager* A_3

$C(3)$ to $C^{(3)}(2)$. The web environment remains inactive. From $A^{(1)}(2)B^{(2)}(2)C^{(3)}(2)$ we have two possibilities to continue. If we apply environmental table P_{E_1}, then we will obtain: $A^{(1)}(2)B^{(2)}(2)C^{(3)}(2) \Longrightarrow_\Gamma ABC$, whereas if we use P_{E_2}, then we will receive: $A^{(1)}(2)B^{(2)}(2)C^{(3)}(2) \Longrightarrow_\Gamma A(3)A^{(1)}(1)B(3)B^{(2)}(1)C(3)C^{(3)}(1)$. ABC is a terminal string. Continuing the derivation with $A(3)A^{(1)}(1)B(3)B^{(2)}(1)C(3)C^{(3)}(1)$, we can employ either environmental table P_{E_1} or table P_{E_2}. Let us assume that we will use environmental table P_{E_1} (the continuation of the derivation for table P_{E_2} may be done analogously, thus it is left to the reader). As a result of the utilization of P_{E_1}, we will attain: $A(3)A^{(1)}(1)B(3)B^{(2)}(1)C(3)C^{(3)}(1) \Longrightarrow_\Gamma A^{(1)}(2)AB^{(2)}(2)BC^{(3)}(2)C \Longrightarrow_\Gamma AABBCC$. We applied again environmental table P_{E_1} at the last step and received a terminal string. The derivation could have been continued in a different way, if we had employed environmental table P_{E_2} at the last step. The verification is left to the reader. Any other choice of the labels of the agents will not lead to a terminal string.

Observe that L_2 is a context–sensitive language, but it is not context–free.

4.4.4 The Power of Eco–Foraging Systems with Time

We will show that the language family determined by unordered scattered context grammars of finite index is equal to the language family generated by programmed eco–foraging systems with time with appearance checking.

Theorem 3 $\mathcal{L}_{fin}(\mathrm{USC}) = \mathcal{L}(\mathrm{FEG}^{time}_{\mathrm{PR}_{ac}})$.

Proof
First, we will prove that $\mathcal{L}_{fin}(\mathrm{USC}) \subseteq \mathcal{L}(\mathrm{FEG}^{time}_{\mathrm{PR}_{ac}})$. Let L be a language generated by an unordered scattered context grammar $G = (N, T, S, P)$ of finite index, where $ind(L) = ind(L(G))$. To verify the statement, we will construct a $\mathrm{FEG}^{time}_{\mathrm{PR}_{ac}}$ system Γ, such that $L(\Gamma) = L(G)$.

Let $P = \{p_1, p_2, \ldots, p_s\}$, where $p_l : (X_{l,1} \rightarrow x_{l,1}, \ldots, X_{l,k_l} \rightarrow x_{l,k_l})$, $1 \leq l \leq s, k_l \geq 1$. Let us assume that $ind(L(G)) = r$ and $r \geq k_l$. Then for all $\omega \in L(G)$, there exists a derivation

$S \Longrightarrow \omega_1 \Longrightarrow \omega_2 \ldots \Longrightarrow \omega_z = \omega, z \geq 1$, such that there are at most r nonterminal symbols in $\omega_q, 1 \leq q \leq z$. These are the derivations that we will simulate.

Before we give the details of the simulation, we make some observations about the derivations of finite index in G.

Let $\omega = u_1 D_1 u_2 \ldots u_j D_j u_{j+1}$ be a sentential form in G, where $j \leq r$, $D_k \in N$, $u_h \in T^*$, $1 \leq k \leq j$, $1 \leq h \leq j+1$. Let us call $[D_1 \ldots D_j]$ the nonterminal cut of ω, and let us denote the nonterminal cut by $c^{(nt)}(\omega)$. We regard nonterminal cuts $[D_1 \ldots D_j]$ as equivalent with respect to all permutations of letters $D_1, D_2, \ldots, D_j$.

Let us denote by $\mathcal{C}$ the set of all nonterminal cuts of the sentential forms in G. Let $\mathcal{C}_r$ be the set of elements of $\mathcal{C}$ of length at most r. By the length of the nonterminal cut of $\omega \in (N \cup T)^*$, we mean the number of nonterminals in $c^{(nt)}(\omega)$.

We say that nonterminal cut $c'_1 = [D_1 \ldots D_j]$ yields nonterminal cut $c'_2 = [B_1 \ldots B_l]$, $1 \leq j \leq r$, $0 \leq l \leq r$, through the use of rule $p \in P$, denoted by $c'_1 \mapsto_p c'_2$, if there are two sentential forms ω_1 and ω_2 in G, such that $c^{(nt)}(\omega_1) = c'_1$, $c^{(nt)}(\omega_2) = c'_2$ and if we apply rule p to ω_1, then $\omega_1 \Longrightarrow_G \omega_2$ holds. We can determine the set of rules $P'_{c'_1,c'_2} \subseteq P$ for arbitrary two nonterminal cuts $c'_1 = [D_1 \ldots D_j]$ and $c'_2 = [B_1 \ldots B_l]$, where $1 \leq j \leq r$, $0 \leq l \leq r$, such that for any $p \in P'_{c'_1,c'_2}$, $c'_1 \mapsto_p c'_2$ holds. (Notice that this rule set can be empty.)

Let $\mathcal{D} = \{(c'_1, c'_2, p) \mid c'_1 \mapsto_p c'_2, p \in P, c'_1, c'_2 \in \mathcal{C}_r\}$. Since the number of nonterminals in the nonterminal cuts is bounded by r and the number of productions in G is a finite set, $\mathcal{D}$ is a finite set, as well.

Observe that if $S \Longrightarrow_{p_{i_1}} \omega_1 \Longrightarrow_{p_{i_2}} \omega_2 \Longrightarrow_{p_{i_3}} \ldots \Longrightarrow_{p_{i_z}} \omega_z = \omega, z \geq 1$, is a derivation in G, such that there are at most r nonterminal symbols in $\omega_q, 1 \leq q \leq z$, then $c^{(nt)}(S) \mapsto_{p_{i_1}} c^{(nt)}(\omega_1) \mapsto_{p_{i_2}} c^{(nt)}(\omega_2) \mapsto_{p_{i_3}} \ldots \mapsto_{p_{i_z}} c^{(nt)}(\omega_z) = c^{(nt)}(\omega)$ holds. Furthermore, $c^{(nt)}(S) \mapsto_{p_{i_1}} c^{(nt)}(\omega_1) \mapsto_{p_{i_2}} c^{(nt)}(\omega_2) \mapsto_{p_{i_3}} \ldots \mapsto_{p_{i_z}} c^{(nt)}(\omega_z) = c^{(nt)}(\omega)$ may belong to several derivations in G. It signifies that starting from S and applying the rules $p_{i_1}, \ldots, p_{i_z}$ in this order to the corresponding sentential forms, we obtain all derivations $S \Longrightarrow_{p_{i_1}} u_1 \Longrightarrow_{p_{i_2}} u_2 \Longrightarrow_{p_{i_3}} \ldots \Longrightarrow_{p_{i_z}} u_z = u$ in G, where $c^{(nt)}(u_j) = c^{(nt)}(\omega_j)$, $1 \leq j \leq z$, holds.

In order to prove that $\mathcal{L}_{fin}(\mathrm{USC}) \subseteq \mathcal{L}(\mathrm{FEG}^{time}_{\mathrm{PR}_{ac}})$, we construct $\Gamma \in \mathrm{FEG}^{time}_{\mathrm{PR}_{ac}}$, such that

for each triplet (c'_1, c'_2, p) in $\mathcal{D}$, the foragers of Γ indicate how the the nonterminals will be replaced, if $c'_1 \mapsto_p c'_2$, and only that. These foragers may identify not only the nonterminals rewritten by p in the sentential form, but also (depending on p) those nonterminals that remain unaltered. To complete the simulation of the application of p, the environment substitutes (some of) the nonterminals and may perform other kinds of rewritings. The idea is that every derivation $S \Longrightarrow_{p_{i_1}} \omega_1 \Longrightarrow_{p_{i_2}} \omega_2 \Longrightarrow_{p_{i_3}} \ldots \Longrightarrow_{p_{i_z}} \omega_z = \omega$ in G corresponds to a computation $U_{i_0} M_{i_0}(t_{i_0}) S(t_0) \Longrightarrow^* U_{i_1} M_{i_1}(t_{i_1}) \omega_1(t_1) \Longrightarrow^* U_{i_2} M_{i_2}(t_{i_2}) \omega_2(t_2) \Longrightarrow^* \ldots \Longrightarrow^* U_{i_{z-1}} M_{i_{z-1}}(t_{i_{z-1}}) \omega_{i_{z-1}}(t_{z-1}) \Longrightarrow^* U_{i_z} \omega_z(t_z) = U_{i_z} \omega(t_z)$ in Γ and vice versa, where $M_{i_h}(t_{i_h})$, $t_{i_h} \in \mathbb{N}$, $0 \leq h \leq z-1$, is the starting nonterminal of the forager commencing the simulation of the application of production p_{i_h} to $w_{z_{i_{h-1}}}$ and $t = r + 1$ (r is the index of G).

Now we define the components of Γ. As in the proof of Theorem 2, we present the foragers and the environmental tables with which these foragers cooperate during the simulation.

Let $card(\mathcal{D}) = m$ and let $M_k = (c'_1, c'_2, p) \in \mathcal{D}$, $1 \leq k \leq m$, where nonterminal cut $c'_1 = [D_1 \ldots D_j]$ yields nonterminal cut $c'_2 = [B_1 \ldots B_l]$, $1 \leq j \leq r$, $0 \leq l \leq r$, if we use rule $p \in P$.

For M_k, $k \in \{i_1, \ldots, i_z\}$, we construct foragers $A_{k,i}$, $1 \leq i \leq j$. Let $A_{k,i} = (\bar{N}_{k,i} \cup N^{(k,i)}_{k,i}, S_{k,i}, R_{k,i})$, where $\bar{N}_{k,i} = \{M_{k,i,0}(t), D_i(t') \mid 1 \leq t \leq t_{k,i,0}, 1 \leq t' \leq t_{k,i,1}\}$, $N^{(k,i)}_{k,i} = \{M^{(k,i)}_{k,i,0}(t), D^{(k,i)}_i(t') \mid 0 \leq t \leq t_{k,i,0}, 0 \leq t' \leq t_{k,i,1}\}$, and $S_{k,i} = M_{k,i,0}(t_{k,i,0})$, $t_{k,i,0} \geq 1$. The rule set of forager $A_{k,i}$, i.e. $R_{k,i}$, $1 \leq i \leq j$, is as follows:

- $(l_{k,i,0}(t) : M_{k,i,0}(t) \rightarrow M^{(k,i)}_{k,i,0}(t-1), l_{k,i,1}, l_{k,i,0})$, $1 \leq t \leq t_{k,i,0}$,
- $(l_{k,i,1}(t) : D_i(t) \rightarrow D^{(k,i)}_i(t-1), l_{k,i,0}, h_{k,i,1})$, $1 \leq t \leq t_{k,i,1}$,
- $(h_{k,i,1}(t) : D^{(k,i)}_i(t) \rightarrow D^{(k,i)}_i(t-1), l_{k,i,1}, h_{k,i,1})$, $1 \leq t \leq t_{k,i,1}$, where
  - $l_{k,i,0} = \{l_{k,i,0}(t) \mid 1 \leq t \leq t_{k,i,0}\}$, $l_{k,i,1} = \{l_{k,i,1}(t) \mid 1 \leq t \leq t_{k,i,1}\}$, $h_{k,i,1} = \{h_{k,i,1}(t) \mid 1 \leq t \leq t_{k,i,1}\}$.

For forager $A_{k,i}$, $1 \leq i \leq j-1$, simulating derivation $U_{k,i,0} M_{k,i,0}(t_{M_{k,i,0}}) \omega(t_\omega)$ $\Longrightarrow^* U_{k,i+1,0} M_{k,i+1,0}(t_{M_{k,i+1,0}}) \omega'(t_{\omega'})$, we will construct the environmental tables. For technical reasons, let $P_E^{\{a,F\}} = \{a \rightarrow a \mid a \in T_E\} \cup \{F \rightarrow F\}$.

The first environmental table $P_{k,i,0}$, $1 \leq i \leq j-1$, is of the form below:

$$\begin{aligned} P_{k,i,0} =& \{U_{k,i,0} \to U_{k,i,1}, U' \to F \mid U' \in V_M \setminus \{U_{k,i,0}\}\} \cup \\ & \{C(t) \to C(t-1), C(0) \to F \mid C \in V_E \setminus (V_M \cup T_E), t \in \mathbb{N}\} \cup P_E^{\{a,F\}}. \end{aligned}$$

By the joint action of forager $A_{k,i}$, $1 \leq i \leq j-1$, $(M_{k,i,0}(t) \to M_{k,i,0}^{(k,i)}(t-1),\ t \in \mathbb{N})$ and the environmental table, $P_{k,i,0}$, we obtain $U_{k,i,0}M_{k,i,0}(t_{M_{k,i,0}})\omega(t_\omega) \Longrightarrow U_{k,i,1}M_{k,i,0}^{(k,i)}(t_{M_{k,i,0}} - 1)\omega(t_\omega - 1)$.

At the second step, forager $A_{k,i}$, $1 \leq i \leq j-1$, identifies D_i in the string. The second environmental table has the form:

$$\begin{aligned} P_{k,i,1} =& \{U_{k,i,1} \to U_{k,i+1,0}, U' \to F \mid U' \in V_M \setminus \{U_{k,i,1}\}\} \cup \\ & \{C(t) \to C(t-1), C(0) \to F \mid C \in V_E \setminus (V_M \cup T_E \cup \{M_{k,i,0}^{(k,i)}(t'-1)\}), t, t' \in \mathbb{N}\} \cup \\ & \{M_{k,i,0}^{(k,i)}(t) \to M_{k,i+1,0}(t'), M_{k,i,0}^{(k,i)}(0) \to F \mid t, t' \in \mathbb{N}\} \cup P_E^{\{a,F\}}. \end{aligned}$$

At the second step, forager $A_{k,i}$, $1 \leq i \leq j-1$, performs rule $D_i(t) \to D_i^{(k,i)}(t-1)$, $t \in \mathbb{N}$. As a consequence of the parallel action of the forager and the environment, we receive the word $U_{k,i,1}M_{k,i,0}^{(k,i)}(t_{M_{k,i,0}}-1)\omega(t_\omega-1) \Longrightarrow U_{k,i+1,0}M_{k,i+1,0}(t_{M_{k,i+1,0}})\omega'(t_\omega-2)D_i^{(k,i)}(t_\omega-2)\omega''(t_\omega-2)$.

If $i = j$, then the first step of the simulation is analogous to the case when $A_{k,i}$, $1 \leq i \leq j-1$, acts in parallel with environmental table $P_{k,i,0}$. The second environmental table, $P_{k,j,1}$, should be modified as follows:

$$\begin{aligned} P_{k,j,1} =& \{U_{k,j,1} \to \hat{U}_{k',1,0}, U' \to F \mid U' \in V_M \setminus \{U_{k,j,1}\}, k' \in \{i_1, \ldots, i_z\}, k' \neq k\} \cup \\ & \{C(t) \to C(t-1), C(0) \to F \mid C \in V_E \setminus (V_M \cup T_E \cup \{M_{k,j,0}^{(k,j)}(t'-1)\}), t, t' \in \mathbb{N}\} \cup \\ & \{M_{k,j,0}^{(k,j)}(t) \to \lambda, M_{k,j,0}^{(k,j)}(0) \to F \mid t \in \mathbb{N}\} \cup P_E^{\{a,F\}}. \end{aligned}$$

After the second step through the parallel action of forager $A_{k,j}$, $(D_j(t) \to D_j^{(k,j)}(t-1)$,

$t \in \mathbb{N}$) and environmental table, $P_{k,j,1}$, we obtain the word $U_{k,i,1}M^{(k,j)}_{k,j,0}(t_{M_{k,j,0}} - 1)\omega(t_\omega - 1) \Longrightarrow \hat{U}_{k',1,0}\omega'(t_\omega - 2)D^{(k,j)}_j(t_\omega - 2)\omega''(t_\omega - 2)$.

After forager $A_{k,j}$ has finished the substitution of $D_j(t)$ for $D^{(k,j)}_j(t-1)$, $t \in \mathbb{N}$, and the corresponding environmental has replaced all the other symbols, then only the environment is allowed to continue the derivation due to the lack of symbols M_k. At this step, the environment replaces the nonterminals of the nonterminal cut present at the environment word and identified by the foragers and it either starts the simulation of the next possible configuration transmission or finishes the derivation. The environmental table can be defined as follows:

$$
\begin{aligned}
P_{k'} =& \{\hat{U}_{k',1,0} \to U_{k',1,0}M_{k',1,0}(t), \hat{U}_{k',1,0} \to \lambda \mid t \in \mathbb{N}\} \cup \\
& \{U' \to F \mid U' \in V_M \setminus \{\hat{U}_{k',1,0}\}\} \cup \\
& \{C(t) \to C(t-1) \mid C \in V_E \setminus (V_M \cup T_E \cup \{D^{(k,1)}_1(t_1), \ldots, D^{(k,j)}_j(t_j)\}), t, t_1, \ldots, t_j \in \mathbb{N}\} \cup \\
& \{C(0) \to F \mid C \in V_E \setminus (V_M \cup T_E \cup \{D^{(k,1)}_1(t_1), \ldots, D^{(k,j)}_j(t_j)\}), t, t_1, \ldots, t_j \in \mathbb{N}\} \cup \\
& \{D^{(k,1)}_1(t_1) \to \alpha_1, \ldots, D^{(k,j)}_j(t_j) \to \alpha_j \mid \alpha_1, \ldots, \alpha_j \in (V_E \setminus V_M)^*, t_1, \ldots, t_j \in \mathbb{N}\} \cup \\
& P^{\{a,F\}}_E.
\end{aligned}
$$

As a result of the employment of $P_{k'}$, the environmental word may have either the form $\hat{U}_{k',1,0}\omega'(t_{\omega'})D^{(k,1)}_1(t_1)\ldots D^{(k,j)}_j(t_j)\omega''(t_{\omega''}) \Longrightarrow U_{k',1,0}M_{k',1,0}(t)\omega'(t_{\omega'} - 1)\alpha_1 \ldots \alpha_j\omega''(t_{\omega''} - 1)$ or the form $\hat{U}_{k',1,0}\omega'(t_{\omega'})D^{(k,1)}_1(t_1)\ldots D^{(k,j)}_j(t_j)\omega''(t_{\omega''}) \Longrightarrow \omega'(t_{\omega'} - 1)\alpha_1 \ldots \alpha_j\omega''(t_{\omega''} - 1)$.

It only remains to be shown how the simulation begins. The initial state of the web environment is $U_{i_0}M_{i_0}(t_{i_0})S(t_0)$. We have to simulate rule $S \to \omega_1 \in P$. Let us assume that the web environment has a table $P_{E_{start}}$ that performs this simulation, where

$$
\begin{aligned}
P_{E_{start}} =& \{S(t_0) \to \omega_1(t_1) \mid t_0, t_1 \in \mathbb{N}\} \cup \{U_{i_0} \to U_{i_1}, U' \to F \mid U' \in V_M \setminus \{U_{i_0}\}\} \cup \\
& \{M_{i_0}(t_{i_0}) \to M_{i_1}(t_{i_1}) \mid t_{i_0}, t_{i_1} \in \mathbb{N}\} \cup \\
& \{C(t) \to F \mid C \in V_E \setminus (V_M \cup T_E \cup \{M_{i_0}(t_{i_0})\}), t, t_{i_0} \in \mathbb{N}\} \cup P^{\{a,F\}}_E.
\end{aligned}
$$

Initially, all the foragers are inactive, since there is not a $M_{i_h}, 1 \leq h \leq z$, in the sentential form. The derivation is as follows: $U_{i_0}M_{i_0}(t_{i_0})S(t_0) \Longrightarrow U_{i_1}M_{i_1}(t_{i_1})\omega_1(t_1)$. Since S does not occur in any other rule of G, this is the only way how the simulation can begin.

During the simulation the sequences $(c_1', c_2', p) \in \mathcal{D}$ determine the valid configuration transmissions. Since the agents belonging to a nonterminal cut rewrite all of it nonterminals and only those, therefore it can be concluded that we can simulate all of the appropriate derivations in G. If the environmental word contains only terminals, then it is the element of the generated language, otherwise it is not. Hence the inclusion $\mathcal{L}_{fin}(\mathrm{USC}) \subseteq \mathcal{L}(\mathrm{FEG}^{time}_{\mathrm{PR}_{ac}})$ is verified.

Secondly, we will prove the other direction, i.e. $\mathcal{L}_{fin}(\mathrm{USC}) \supseteq \mathcal{L}(\mathrm{FEG}^{time}_{\mathrm{PR}_{ac}})$. Let $\Gamma = (E, A_1, \ldots, A_n, c_{init})$ be an $\mathrm{FEG}^{time}_{\mathrm{PR}_{ac}}$ system, defined as in Def. 21. To prove the statement, we will construct an unordered scattered context grammar $G = (N, T_E, P, S)$ of finite index, such that $L(\Gamma) = L(G)$ holds. The proof is based on simulating the derivations in Γ with derivations of finite index in G.

Let $c = (k_1, \ldots, k_n; u_1 D_{h_1}(t_1) u_2 D_{h_2}(t_2) \ldots u_j D_{h_j}(t_j) u_{j+1})$ be a configuration of Γ, where $t_j \in \mathbb{N}_0$, $j \geq 1$, $D_{h_m} \in V_N \cup \bar{V}_N$, $1 \leq m \leq j$, $u_l \in T_E^*$, $1 \leq l \leq j+1$, and $k_1, \ldots, k_n$ are the labels of the rules to be applied by the agents. For the sake of legibility, we also refer to the elements of $V_N \cup \bar{V}_N$ as the nonterminals of Γ. Observe that there must be a number s, $s \in \mathbb{N}$, such that if there are more than s nonterminals in the environmental state $\omega = u_1 D_{h_1}(t_1) u_2 D_{h_2}(t_2) \ldots u_j D_{h_j}(t_j) u_{j+1}$, i.e. $j > s$, then after a certain number of derivation steps, some of the t_js will become 0. This statement can be explained by the fact that if there are n agents in the eco–foraging system and t is the maximum of the lifetimes associated with the nonterminals of Γ, and there are $n \cdot t + 1$ such nonterminals in the environmental state v of some configuration, then after performing $t+1$ derivation steps on environmental state v, we will obtain an environmental state containing at least one nonterminal whose lifetime is equal to 0.

Owing to the fact that it is not possible to remove the nonterminal with lifetime 0 from the strings, every derivation in Γ that results in a word over T_E cannot contain a configuration in which the environmental state has a nonterminal with lifetime 0. Consequently, when we simulate the derivations of Γ with derivations of G, it is enough to consider derivations with configurations of the forms $c = (k_1, \ldots, k_n; u_1 D_{h_1}(t_1) u_2 D_{h_2}(t_2) \ldots u_j D_{h_j}(t_j) u_{j+1})$ only, where D_{h_m} is a nonterminal, $1 \leq m \leq j$, $u_r \in T_E^*$, $t_j \in \mathbb{N}$, $1 \leq r \leq j+1$, and $j \leq s$. Analogously to the first part of the proof, let us call $c^{(nt)}(c) = [D_{h_1}(t_1) D_{h_2}(t_2) \ldots D_{h_j}(t_j)]$

the nonterminal cut of c.

Since $s \in \mathbb{N}$ and the set of nonterminals as well as the set of rules of the agents are finite sets, we can determine the set of all valid configuration transmissions. The configuration transmission from c'_1 to c'_2 is valid, provided that the lengths of their nonterminal cut are less than s and there is no nonterminal in the nonterminal cuts with lifetime 0.

Let us label elements of $\mathcal{P}_E$ by elements of $Label(\mathcal{P}_E)$ and let us suppose that $Label(\mathcal{P}_E)$ and $Label(R_i)$, $1 \leq i \leq n$, are pairwise disjoint sets.

As before, we say that nonterminal cut $c'_1 = [D_1 \dots D_j]$ yields nonterminal cut $c'_2 = [B_1 \dots B_l]$, $1 \leq j \leq s$, $0 \leq l \leq s$, through the use of the rules labelled by $\bar{k}_1, \dots, \bar{k}_j$, where $\bar{k}_1, \dots, \bar{k}_j \in Label(\mathcal{P}_E) \cup \bigcup_{i=1}^{n} Label(R_i)$ denoted by $c'_1 \mapsto_{(\bar{k}_1,\dots,\bar{k}_j)} c'_2$, if there are two configurations c_1 and c_2 in Γ, such that $c^{(nt)}(c_1) = c'_1$, $c^{(nt)}(c_2) = c'_2$, and if we apply the rules labelled by $\bar{k}_1, \dots, \bar{k}_j$ to c_1, then $c_1 \Longrightarrow_\Gamma c_2$ holds. We can determine the labels of the rules of all such possible rule sets.

We can construct the finite set $\mathcal{D} = \{[(k_1, \dots, k_n); (\bar{k}_1, \dots, \bar{k}_j); (c'_1, c'_2); (k'_1, \dots, k'_n)] \mid c'_1 = c^{(nt)}(c_1), c'_2 = c^{(nt)}(c_2), c'_1 \mapsto_{(\bar{k}_1,\dots,\bar{k}_j)} c'_2$, c_1 and c_2 are configurations of Γ, $\bar{k}_h \in Label(\mathcal{P}_E) \cup \bigcup_{i=1}^{n} Label(R_i), 1 \leq h \leq j, k_v, k'_v \in Label(R_v), 1 \leq v \leq n, k_v$ is the current label of R_v, and k'_v is the next label of $R_v\}$. Note we can employ this construction only because the terminal symbols are not altered and the number of nonterminals in each environmental state we consider is limited by a constant. Furthermore, since we know the current labels of the rules the agents apply, we can calculate the new labels of the rules of the agents, as well.

Based on the above observations, G have the rules of the following form:

- $(S \to M_{init}\omega_{init})$, where ω_{init} is the axiom of Γ,
  $M_{init} = [(\emptyset, \dots, \emptyset); (\emptyset, \dots, \emptyset); (\emptyset, c'_{init}); (k_1, \dots, k_n)]$, the eco–grammar system has not started to work in the beginning, we denote this fact by the dummy symbol $(\emptyset)$, which refers to the empty label of the agents and the environment as well as to the empty configuration, $c'_{init} = c^{(nt)}(c_{init})$, and $k_1, \dots, k_n$ are the initial labels of the agents,
- $(M \to M', D_{h_1}(t_1) \to \alpha_1, \dots, D_{h_j}(t_j) \to \alpha_j)$, where $M, M' \in \mathcal{D}$,

$M = [(k_1, \ldots, k_n); (\bar{k}_1, \ldots, \bar{k}_j); (c'_1, c'_2); (k'_1, \ldots, k'_n)]$,
$M = [(k'_1, \ldots, k'_n); (\bar{k'}_1, \ldots, \bar{k'}_l); (c'_2, c'_3); (k''_1, \ldots, k''_n)]$,
and k_m is the label of $D_{h_m}(t_1) \rightarrow \alpha_m$, $1 \leq m \leq j$, $1 \leq j \leq s$, $0 \leq l \leq s$,

- $(M \rightarrow M_{fin}, a \rightarrow a)$, where $a \in T_E$,
- $(M_{fin} \rightarrow \lambda)$.

Using these productions, we can simulate the derivations in Γ, where the number of nonterminals is not more that r and none of the nonterminals has lifetime 0 (see above), by derivations in G. Due to the fact that the rules above follow the order of the configuration transmissions, G generates the same words as Γ does. Hence the theorem is proved. ■

4.5 Experiments

We have conducted an eighteen–day experiment on the web while amassing real data. The crawlers utilize either the weblog algorithm, the RL algorithm or their combination.

4.5.1 Web

The experiment has been performed on a single personal computer. The forager architecture is implemented in Java. During the simulation a fixed number of foragers compete against each other to collect news items from the CNN web site. They crawl along their paths in quest of novel information for equal time intervals according to a predefined order. After the given period has elapsed, the forager has to finish the step that it has started.

The link structure of the collected web pages follows the power–law distribution ($P(k) = k^{\gamma}$): γ equals -1.45 for the outgoing links and -3.04 for the incoming ones, as it is demonstrated in Fig. 4.1. Thus it can be concluded that the link structure has the scale–free property. The clustering coefficient [129] of the link structure is 0.02 and the diameter of the graph is 7.2893. We have reordered the link structure as follows: we have applied

random permutations to the start and end points of the links. The reordered links have the same distribution.

The magnitude of the clustering coefficient of the new generated graph is lower by an order of magnitude than that of the original graph, whereas the diameter of the new graph remains the same. It implies that the links of the gathered pages form a small world structure. The pages for the experiment are stored in a centralized component with two indices (and time stamps): the URL and the page index. Multiple instances of a page may have the same URL index provided that it has been downloaded from the same URL and the content of the page has been changed. The page index, in effect, uniquely identifies the page content and the URL belonging to it. For more details, see [99, 102].

4.5.2 Tasks, Crawlers and the Simulated Web Environment

We have performed the following types of experiments:

1. Experiments without communication:
   (a) There is no communication between the crawlers. The documents are relevant, if they are found within 24 hours of their respective time stamps. In this experiment, the previous results of [99] have been reproduced (+ signs in Fig. 4.2, case *reproduced*).
   (b) Topic–specific experiments without communication ($\circ$ signs in Fig. 4.2, case *no comm*).
2. Topic–specific experiments with communication:
   (a) Both types of document sending, i.e. sending to the RA and sending to other crawlers, are adaptive ($\bigtriangledown$ signs in Fig. 4.2, case *learn all*).
   (b) Sending a document to the RA is adaptive, but the situation is more straightforward: each crawler sends its learned weight to the other crawlers and the crawlers utilize the weights they receive in the direct communication of the documents ($\bigtriangleup$ signs in Fig. 4.2, case *send learned*).

(c) Averaged weights of the previous experiment are used in both types of communication ($\times$ signs in Fig. 4.2, case *good all*).

We have generated three different environments for the simulations:

1. SFSW: each simulated page has exactly the same links as the original page on the web does.
2. SFW: the number of links of the new simulated pages equals that of the original pages on the web. New documents are added according to their time stamps. A new document preserves the number of its links. The targets of these links are selected from the already existing simulated web by the preferential attachment algorithm (see Chapter 2).
3. RWE: similar to the SFW except that the targets of the links of a new document are selected randomly from the already existing simulated web according to the uniform distribution.

The downloaded portion of the environment is a small world and demonstrates the scale–free property (Fig. 4.1), i.e. it is an SFSW environment. The clustering coefficient of the SF environment (Fig. 4.1), on the other hand, is ten times smaller than that of the SFSW environment. Nevertheless, it is also scale–free with respect to both the incoming and the outgoing degree distributions. The clustering coefficient of the RWE environment is also ten times smaller than that of the SFSW environment. The outgoing link degree distribution of this environment is scale–free, whilst the incoming link degree distribution is exponential (Fig. 4.1) owing to the uniform selection of the linked documents.

4.5.3 Simulations

Fig. 4.2 summarizes the results of the simulations. In all cases, the number of the downloaded documents is approximately the same ($\sim 2 \times 10^6$ downloads). Fig. 4.2(a) shows the number of found documents sent to the RA. Fig. 4.2 (b) illustrates the number of

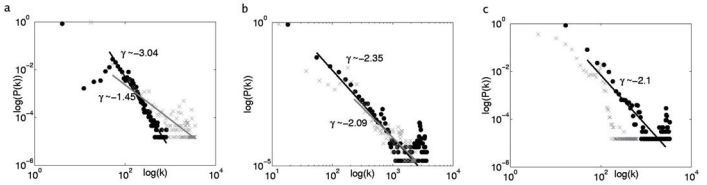

Figure 4.1: **Scale–free properties of the environments.** Log-log scale distribution of the number of (incoming and outgoing) links of all URLs found during the investigation. Horizontal axis: number of edges ($\log k$). Vertical axis: relative frequency of the number of edges at different URLs ($\log P(k)$). The dots and the dark line correspond to the outgoing links, the crosses and the gray line to the incoming links: (a) the downloaded portion of the Internet (SFSW environment); (b) rewiring of the outgoing links of the new nodes by the preferential attachment algorithm (SFW environment); (c) the random rewiring of the outgoing links of the new nodes (RWE). For further details, see the text.

documents found by the crawlers, sent to the RA and reinforced by the RA. Fig. 4.2(c) depicts the number of found documents received by the crawlers. Fig. 4.2(d) demonstrates the number of documents received by the crawlers, forwarded to the RA and reinforced by the RA. Fig. 4.2(e) and (f) show the sums of values of Figs. 4.2(a) and (c), and Figs. 4.2(b) and (e), respectively.

4.6 Discussion and Conclusions

In this chapter we have presented a theoretical as well as a practical approach to the behaviour of Internet crawlers seeking novel information on the World Wide Web. In the sequel, we give a brief review of our main achievements and present some issues to be elaborated in the context of web crawlers from both points of view.

4.6.1 Eco–Foraging Systems

First, we have proposed programmed grammars, or more precisely, programmed grammar schemes, for the description of the behaviour of the crawlers. Our aim is to illustrate the great diversity of employing regulated rewriting devices in the field of web crawling techniques. Evidently, other benefits could be reaped as a result of the formalization of the web spidering by means of grammars with other types of controlled derivations [36], [110]. Besides programmed grammars, there are other regulated devices that prescribe the sequences of productions or determine the dependence of the rule on the history of the derivation. The idea of prescribing the sequences of productions or determining the dependence of the rule on the history of the derivation can correspond to the utilization of some kind of ordering in the case of URLs. Grammars that impose some global context condition on the employment of productions can be suitable candidates for capturing the behaviour of focused/topic–specific/topical crawlers. Contrary to grammars with prescribed sequences or with dependence of the rule on the history of the derivation, these regulated rewriting mechanisms do not determine the sequence of applicable productions in advance, since it is controlled by the generated sequence of the sentential forms, i.e. by the environmental string produced through the joint actions of the foragers and the web environment. Crawlers may seek information either on a single (identical) topic, or on different ones [78]. The idea of information harvest on similar topics can be expressed by regulated rewriting mechanisms employing some sort of parallelism. Some regulated rewriting mechanisms may also be combined with another. In this way, several different aspects of search strategies can be formalized and more sophisticated techniques can be realized. Furthermore, certain regulated rewriting devices may be incorporated into Lindermayer systems.

Secondly, in our work the evolution of the web environment is determined by means of interactionless Lindenmayer systems. The structure of the World Wide Web, however, demonstrates the scale–free small world property [15, 84, 129]. The emergence of power–law behaviour of the out– and in–degree distributions results in the occurrence of temporal fractal structures [44, 83] in the World Wide Web [123]. In reality, (stochastic) parametric interactionless Lindenmayer systems [110] are particularly convenient to describe the growth of the World Wide Web. Therefore a further refinement of the model could be to replace interactionless Lindenmayer systems with (stochastic) parametric ones. The in-

vestigation of the characteristics as well as the generative power of eco–grammar systems consisting of various regulated rewriting devices are the subjects of further research.

In the third place, in our model the foragers produce observable output signals in reply to the stimuli received. Consequently, it is a natural way to model the crawlers as finite state machines [58, 110, 112]. Nonetheless, in practice unexpected inputs might emerge, which means that in this respect they behave rather like stochastic machines. A further direction could be to include some kind of randomness in the model. The crawlers may be formalized with the aid of probabilistic grammars and weighted grammars [111, 112]. The application of the latter proposition has been already investigated in the case of cooperating grammar systems [6, 29] and of parallel communicating grammar systems [5].

Finally, the crawlers are able to learn through gathering valuable pieces of information. In effect, to describe the learning process the finite state description can become inadequate in some cases. The possible state space is infinite, though, the machine works on a finite, forever changing set of states. The set of states depends on the learning algorithm [54, 81]. In addition, the state transitions may also be altered. While the crawlers learn, they may be organized into groups, or in other words, into fleets. In the goal–oriented setting, the crawlers tend to work in different compartments, or niches: the crawlers avoid the overlaps between their paths. In this way labour division might be attained [78, 99, 102].

4.6.2 Experiments

In our experiments we have examined the extent to which communication makes a goal–oriented community efficient in different topologies. To this end, we have studied the influence of learning method as well as that of the topology of the environment on the communication efficiency of crawlers seeking novel information on different topics on the Internet. On the basis of our results, it can be concluded that the performance of different methods may differ considerably in different environments.

First, we have reproduced the previous results of [99] (sign +): in SFSW the selective weblog (WL) algorithm is superior to RL, the addition of RL to the WL algorithm (case WL+RL) results in the deterioration of the performance. On the contrary, combining the

WL algorithm with RL does not reduce the performance of WL in SFW. The crawlers accomplish their task in the SFSW the most, whereas in RWE environment in the least easily on the assumption that there is no communication between them and time constraints are applied.

In the topic–specific cases, studies are restricted by the sparsity of data and time limits are not imposed: all novel documents are rewarded, irrespective of their time stamps. Let us consider first the case without communication (sign $\circ$). RL demonstrates the same tendency: RL has been proven to be much worse than WL in SFSW, nonetheless, the addition of RL to WL (case WL+RL) barely decreases the performance of WL. On the other hand, in SFW WL+RL defeats RL and RL beats WL. In RWE, it is also RL+WL that performs the best. Indeed, the combined method collects twice as much reward as in SFW.

Secondly, if the crawlers are allowed to communicate, then the realization of the task is the most facile in RWE and the most difficult in SFSW, i.e. in the original database, despite the fact that presumably the SFSW environment is not randomly organized (Fig. 4.2(f)). Furthermore, in all worlds, the combined algorithm is the best. The learning and the communication of the weights (sign $\triangle$) are definitely advantageous, though the averaged weights perform similarly well in SFW (sign $\times$). If the crawlers do not communicate their weights, nor are the averaged weights available, then the sending weights can be learned provided that (a portion of) the reward received from the RA is transferred to the sender. This extra learning spoils the performance somewhat (sign $\triangle$, see (Fig. 4.2(c) and (d))). In effect, learning of these weights (i) is not only beneficial in most worlds, (ii) but the performance is also slightly better than the performance when the averaged weights are used (sign $\times$). The weblog algorithm is an extreme case in SFSW, the extra learning of the sending weights augments only sightly the number of collected documents.

Finally, the comparison of different worlds to each other in the topic–specific case is difficult, since the topics are well–organized on the original news site and the organization will be ruined, if the topology is altered. Moreover, the comparison of topic–specific and the non topic–specific cases is an arduous work, as well, by reason of the employment a time limit in the latter one. In fact, novel documents become obsolete, if the time limit has elapsed.

It cannot be deduced from our experiments to what degree the lack of time constraint influences the efficiency of topic–specific crawlers in RWE. Further studies are required to reveal the cause of the high performance in topic–specific cases. A possible explanation is the smallness of the overlaps between paths, i.e. the compartmentalization may be more pronounced in the topic–specific case.

To sum up, it can be concluded that selective learning can be improved by reinforcement learning with and without communication. The SFSW is the only exception, on account of the fact that RL spoils the efficiency of the selective algorithm in the absence of communication. This phenomenon is explicated in [99, 102] by the frequent occurrence of compartmentalization (niche formation) in SFSW. The extension to the topic–specific searches (to environments that require work sharing) and to the cases with communication diminishes the advantages of selective learning in all worlds, including SFSW.

4.6.3 Outlook

As we have shown in this chapter, the structure of the environment may influence significantly the efficiency of the collaboration of the agents. The SFSW has proven to be important. The hierarchical organization of the agents may result in the decrease of computational and communication load. In Chapter 3, the apprentices employ a special peer list for the selection of reliable peers of their masters. Each apprentice may record not only the fact whether a peer engaging in communication with its master is trustworthy or not, but also the degree of the reliability of the peer. In this respect, the list maintained by the apprentices may be regarded as the weblog as we have proposed at the beginning of this chapter. The list belonging to the apprentice, analogously to the weblog of the URLs, contains the peers in a descending order of their weblog values. From the point of view of collaboration, the higher the weblog value of a peer the higher its reliability. If a master wishes to engage in communication with another peer, then the request will be sent to the first few peers from the list updated periodically by its apprentice. The weblog value of a peer will be augmented in case the collaboration between the initiator and the given peer has been successful. On the other hand, the weblog value of a peer will be reduced, if the accomplishment of a task either fails or is delayed. In case the delay is unintended,

then a peer may agree to process a task with a given latency and restart time and it can decide to divide the task into smaller subtasks. The generalization of the one–step buffer conception to a multi–step one suggested in Chapter 3 serves as a viable solution, since it realizes hierarchical task processing by taking into account the previous experiences and exploiting more efficiently the available resources than pre–wired protocols.

The apprentices introduce hierarchy in P2P networks: the master is an element of a hierarchy, its apprentices are one level below in the hierarchy. The systems in which the continuous testing and function of the agents are prescribed by formal language theoretical constraints, may prove to be beneficial from the point of view of scalability, regardless of the fact whether the change is an increase in size or a transformation between the network structures (regular, random, SW, SFW, SFSW).

In the next chapter, we will present a system in which the creation of groups occurs dynamically: after the time dedicated to computation has elapsed, the cluster center will be selected. The cluster center may be regarded as the master, the other agents belonging to the same cluster are those that reply back in a timely manner to the master. We will study the classes of languages that these agents are able to produce. From the underlying language class, we can deduce the difficulty of the level of task the agents are able to solve.

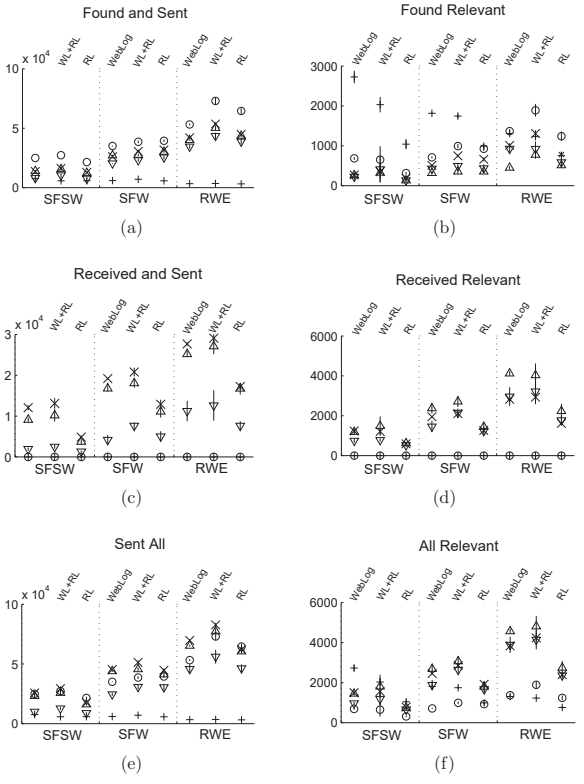

Figure 4.2: **Results in different worlds, with and without communication and with different learning mechanisms.** Notations: +: *reproduced*, ○: *no comm*, ▽: *learn all*, △: *send learned*, and ×: *good all*, SFSW: scale–free small world, SFW: scale–free world, RWE: random world environment, WL: crawling applying the weblog algorithm, RL: crawling utilizing reinforcement learning, WL + RL: the weblog algorithm employs good restarts, the crawling uses RL. Vertical lines: standard deviations of 12 simulations for each case. For more details, see the text.

Chapter 5

Bottom–Up Clustering Algorithm

Communities of interacting and communicating agents create dynamically evolving network topologies. The study of the properties of complex networks is the goal of current research [95]. In particular, a plethora of networks exhibit a small average distance between vertices, a phenomenon called small world effect, typical of random graphs [16], coupled with local clustering properties, characteristic of ordered lattices [129]. Moreover, many random networks are scale–free [15], in the sense that they exhibit a power–law distribution of the degree.

In Chapter 4, on the basis of simulations we have investigated the behaviour of crawlers applying either the selective learning mechanism, the reinforcement learning algorithm or their combination in various network topologies. The crawlers that use reinforcement learning may get stuck in the clusters or may find their way out of a cluster putting a great deal of effort into the escape. Therefore we have developed the weblog update algorithm, which memorizes the most promising URLs for the search. The weblog update algorithm has inspired the introduction of apprentice peers in P2P networks in Chapter 3. The master and the apprentice belonging to it create some kind of hierarchy in P2P systems. The apprentices promote the efficient collaboration of masters working on the same subtask.

In this chapter we present another approach to agents organized into a hierarchy. We utilize simple eco–grammar systems and extend the conditions of dynamic team constitution in

order to capture the behaviour of these agents. We examine the level of difficulty of the problems that the agents are able to solve. The results of this work were appeared in [76].

The organization of this chapter is as follows. In Section 5.1, we give a brief review of the bottom–up clustering algorithm. In Section 5.2, we present the formal definitions and enlighten the basic notions through some examples. In Section 5.3, we establish the relationships of simple eco–grammar systems formed according to the newly introduced conditions to each other as well as to certain language classes of the Chomsky hierarchy and L systems. In Section 5.4, we prove that any recursively enumerable language can be obtained by simple eco–grammar systems, where the active teams assemble according to different conditions of team constitution. Finally, in Section 5.5, we overview the state–of–the–art literature and provide some food for thought for further research.

5.1 Introduction

The bottom–up clustering algorithm is based on the small world concept and the idea that complex networks have high clustering coefficients [9, 20, 38, 81]. In [84] a unified view is proposed to describe both global and local traits of networks by means of a single measure, the connectivity length. It has a precise meaning in terms of information propagation, since the dissemination of information occurs in a highly efficient manner in networks characterized by small global and local connectivity length.

In the bottom–up clustering algorithm the graph consists of nodes connected by directed incoming and outgoing links not necessarily of unit length. The algorithm uses local message passing, which happens within a predefined time according to the decisions influenced by the properties of the *next neighbours* of each node. A node is regarded as a outgoing neighbour of another node, if the latter has an outgoing link to the former and former answers the queries of the latter within a predefined time. Should this condition not be fulfilled, then the outgoing neighbour of the former node will be deleted from the list of the latter. During the bottom–up clustering algorithm, each node enumerates its outgoing neighbours inquiring them about their outgoing links. The nodes collect the requested links until the time devoted to calculation expires. The slowly responding outgoing neighbours

are deleted from the outgoing lists. The nodes calculate their connectivity length values. Cluster formation commences with the circulation of these values. A cluster will be created provided that it has been decided for all respective nodes whether they belong to the cluster and the cluster center has been chosen. The algorithm terminates locally, if there is no local change in the network structure. The algorithm finishes globally on condition that the structure of the network does not alter at global level.

We formalize the behaviour of agents participating in network cluster formation using simple eco–grammar systems [31]. We extend the conditions of dynamic team constitution proposed in [33]. We emphasize that our model is purely a syntactical one. We examine the boundaries of the model at pure syntactic level. Our choice of simple eco–grammar systems with dynamically formed teams of agents to describe the bottom–up clustering algorithm is motivated by the fact that the entities are very simple autonomous agents with limited capabilities, i.e. memoryless and reactive entities working in a dynamically changing problem space. The transformation of the problem space corresponds to the alternation of the environment. The modification of the environment is the result of the joint actions of the agents and the development of the environment. Through their actions the agents contribute to the solution of the problem. The development of the environment expresses the change of the level of difficulty of the problem. The agents are organized into teams: those that are able to disseminate information at an instant are allowed to participate in the creation of a given team. Teams correspond to clusters in the formal language theoretic construction. In the bottom–up clustering algorithm a predefined time limit determines the period in the course of which information propagation may occur: agents usually terminate the connection with those incapable of responding after the time limit has elapsed. In some cases, the delay is unintended, in others, however, it is purposeful. There are different approaches in the literature dealing with how to prevent perpetrators from ruining the function of a system from a formal language theoretic point of view [73, 77].

5.2 Formal Definitions

In the sequel, we give an approach to the bottom–up clustering algorithm in terms of simple eco–grammar systems with dynamically formed teams of agents. Before starting to study the properties of the formal language theoretic constructions to be presented herein, we recall the notion of simple eco–grammar systems from Chapter 2.

Definition 25 *A simple eco–grammar system (a* SEG *system) with n agents, $n \geq 1$, is a construct*

$$\Gamma = (V_E, P_E, R_1, \ldots, R_n, \omega),$$

where

- V_E *is a finite alphabet, the alphabet of the system,*
- P_E *is a finite and complete set of pure context–free rules over V_E (i.e. rules of the form $a \rightarrow \alpha$ with $a \in V_E$, $\alpha \in V_E^*$, and for each $a \in V_E$, there is a rule $a \rightarrow \alpha$ in P_E), the set of developmental rules of the environment,*
- $R_i, 1 \leq i \leq n$, *is a finite set of pure context–free rules over V_E, the set of action rules of the i–th agent,*
- $\omega \in V_E^+$ *is the axiom, the initial state of the environment.*

A string over V_E^* is called the state of the environment or the environmental state. A SEG system functions through the change of its environmental states. The environmental states are altered both by the action rules of the agents and by the developmental rules of the environment.

Remember that $dom(R_i)$ denotes the set of symbols appearing on the left–hand side of the rules of R_i, i.e. $dom(R_i) = \{a \mid a \rightarrow x \in R_i\}$. In fact, $dom(R_i)$ corresponds to the set of symbols that can be modified by an action of the i–th agent.

By a team in a SEG system Γ we mean a set of agents. For a team in a SEG system, we can define two different derivation modes, based on the two versions of parallel rewriting for colonies introduced in [35]. If the *strong derivation mode* is used, then each agent of the given team has to apply one of its productions. It signifies that should an agent be unable to rewrite a symbol of the current sentential form, then the underlying team cannot be employed. In the *weak derivation mode*, however, only those agents that are members of the same team and able to rewrite a symbol of the current sentential form, have to do so. Herein we will use only the strong variant, which we will not indicate separately.

Definition 26 *For a* SEG *system* $\Gamma = (V, P_E, R_1, \ldots, R_n, \omega_E)$, *a team* $\mathcal{T} = \{R_{i_1}, \ldots, R_{i_s}\}$, *and two environmental states* ω, ω', *we define the (strong) direct derivation step (written by* $\omega \models_{\mathcal{T}} \omega'$*) as follows:*

- $\omega_E = x_1 a_1 x_2 \ldots x_s a_s x_{s+1}$ *and* $\omega'_E = y_1 z_1 y_2 \ldots y_s z_s y_{s+1}$, *for some* s, $1 \leq s \leq n$, $a_h \in V_E, x_j, y_j, z_h \in V_E^*$, $1 \leq h \leq s, 1 \leq j \leq s+1$,
- $a_h \to z_h \in R_{i_h}$, $\{i_1, \ldots, i_h\} \subseteq \{1, \ldots, n\}$, $1 \leq h \leq s$,
- $y_j = x_j$ *is either the empty word, or* $x_j \Rightarrow_{P_E} y_j, 1 \leq j \leq s+1$, *is a* 0L *rewriting.*

So as to introduce the extension of the different dynamic team constitution modes proposed in [33], we review the concept of the level of competence/excitation of an agent.

Definition 27 *Let* $\Gamma = (V, P_E, R_1, \ldots, R_n, \omega_E)$ *be a* SEG *system as above. For an environmental state* $\omega \in V^*$, *the level of competence/excitation of* R_i, $1 \leq i \leq n$, *with respect to* ω *is defined as follows:* $lev(R_i, \omega) = card(alph(\omega) \cap dom(R_i))$, *i.e. the number of different symbols from* ω *belonging to* $dom(R_i)$. *We say that* R_i *is competent with respect to* ω, *if* $lev(R_i, \omega) \geq 1$ *holds.*

Informally, the level of competence/excitation of an agent with respect to the environmental state expresses the number of different symbols occurring in the environmental state that can be replaced by that agent.

Definition 28 *Let* $\Gamma = (V, P_E, R_1, \ldots, R_n, \omega_E)$ *be a* SEG *system as above,* $\omega \in V^+$, *and* $\mathcal{T} = \{R_{i_1}, R_{i_2}, \ldots, R_{i_m}\}$, *with* $i_j \in \{1, 2, \ldots, n\}$, $1 \leq j \leq m \leq n$, *be a team of agents in*

Γ, *where each member of* $\mathcal{T}$ *is competent with respect to* ω. *Then* $\mathcal{T}$ *is formed according to condition* $d^{\diamondsuit q}$ *with respect to* ω, $q \in \mathbb{N}_0$, $\diamondsuit \in \{\leq, =, \geq\}$, *if for all* $R_{i_j}, R_{i_k} \in \mathcal{T}$, $i_j, i_k \in \{1, 2, \ldots, n\}$, $1 \leq j, k \leq m \leq n$, $|lev(R_{i_j}, \omega) - lev(R_{i_k}, \omega)| \diamondsuit q$ *and there is no* $R_l, 1 \leq l \leq n$, *such that* R_l *is not an element of* $\mathcal{T}$, R_l *is competent with respect to* ω *and for all members* R_{i_r} *of* $\mathcal{T}$, $i_r \in \{1, 2, \ldots, n\}$, $1 \leq r \leq m \leq n$, $|lev(R_{i_r}, \omega) - lev(R_l, \omega)| \diamondsuit q$ *holds.*

In Def. 28, those agents that are competent with respect to the environmental state and differ from each other in their levels of competence/excitation by at most/exactly/at least q (cases $\leq, =$ and $\geq$) belong to the same team. Observe that singleton teams, i.e. teams consisting of one member, may also be formed and not necessarily one team can satisfy the condition of team constitution in team mode $d^{\diamondsuit q}$, $\diamondsuit \in \{\leq, =, \geq\}$, $q \in \mathbb{N}_0$. Furthermore, in team constitution mode $d^{=q}$, $q \in \mathbb{N}$, the teams can only have two members.

Definition 29 *Let* $\Gamma = (V, P_E, R_1, \ldots, R_n, \omega_E)$ *be a* SEG *system as above,* $\omega \in V^+$, *and* $\mathcal{T} = \{R_{i_1}, R_{i_2}, \ldots, R_{i_m}\}$ *a team of agents in* Γ, *where* $i_j \in \{1, 2, \ldots, n\}$, $1 \leq j \leq m \leq n$, *and each* R_{i_j} *is competent with respect to* ω. *Then* $\mathcal{T}$ *is formed according to condition* $c^{\diamondsuit q}$ *with respect to* ω, *where* $\diamondsuit \in \{\leq, =, \geq\}$, $q \in \mathbb{N}_0$, *if* $card(dom(R_{i_j})) - lev(R_{i_j}, \omega) \diamondsuit q$, $1 \leq j \leq m \leq n$, *and there is no* $R_l, 1 \leq l \leq n$, *such that* R_l *is not an element of* $\mathcal{T}$, R_l *is competent with respect to* ω *and* $card(dom(R_l)) - lev(R_l, \omega) \diamondsuit q$.

Def. 29 could be interpreted as follows: an agent is a member of a given team provided that the agent is competent with respect to the environmental state and the cardinality of the set of symbols appearing on the left–hand side of the rules of the agent differs from its level of competence/excitation by at most/exactly/at least q (cases $\leq, =$ and $\geq$).

Definition 30 *Let* $\Gamma = (V, P_E, R_1, \ldots, R_n, \omega_E)$ *be a* SEG *system as above,* $\omega \in V^+$, $\mathcal{T} = \{R_{i_1}, R_{i_2}, \ldots, R_{i_m}\}$, $i_j \in \{1, 2, \ldots, n\}$, $1 \leq j \leq m \leq n$, *a team of agents in* Γ, *where each member of* $\mathcal{T}$ *is competent with respect to* ω. *Let* $\emptyset \neq V_B, V_C \subseteq V$, $V_B \triangle V_C$, *where* $\triangle \in \{\subseteq, =, \supseteq\}$. *Then* $\mathcal{T}$ *is formed with respect to* ω *according to condition* $t^{\triangle V_B}$, $\triangle \in \{\subseteq, =, \supseteq\}$, *if for all* $R_{i_j} \in \mathcal{T}$, $i_j \in \{1, 2, \ldots, n\}$, $1 \leq j \leq m \leq n$, $(\omega)_{V_C} \in dom(R_{i_j})^+$ *and there is no* $R_l, 1 \leq l \leq n$, *such that* R_l *is not an element of* $\mathcal{T}$, R_l *is competent with respect to* ω *and* $(\omega)_{V_C} \in dom(R_l)^+$ *is satisfied.*

In Def. 30, in case of team constitution mode $t^{=V_B}$, $\emptyset \neq V_B \subseteq V$, the agent is a member of the team, if the agent is competent with respect to the environmental string and the string obtained from the environmental string through the deletion of the letters not belonging to a certain subset V_B, is an element of the set of strings that can be produced using the set of symbols appearing on the left–hand side of the rules of the given agent. Team constitution modes $t^{\subseteq V_B}$ and $t^{\supseteq V_B}$, where $\emptyset \neq V_B \subseteq V$, may be interpreted analogously.

Condition $d^{=0}$ in Def. 28 is the same as condition e, condition $c^{=0}$ in Def. 29 as condition c and condition $t^{=V_B}$, $\emptyset \neq V_B = V$, in Def. 30 as condition t in [33].

Definitions 28, 29 and 30 describe the different cases of cluster formation. A plethora of measures have been proposed in the literature on which cluster creation is based [9, 20, 38, 59, 96].

Definition 31 *For a* SEG *system* $\Gamma = (V, P_E, R_1, \ldots, R_n, \omega_E)$ *as above, and for two environmental states* ω, ω'*, we say that* ω *directly derives* ω' *in* Γ *in team derivation mode* α*, where* $\alpha \in \{d^{\diamondsuit q}, c^{\diamondsuit q}, t^{\triangle V_B} \mid \diamondsuit \in \{\leq, =, \geq\}, \triangle \in \{\subseteq, =, \supseteq\}, q \in \mathbb{N}_0, \emptyset \neq V_B \subseteq V\}$*, denoted by* $\omega \overset{\alpha}{\Longrightarrow}_\Gamma \omega'$*, if one of the following holds:*

- *either* $\omega \models_{\mathcal{T}} \omega'$ *for some team* $\mathcal{T}$ *formed according to condition* α *in* Γ,
- *or, if such a team does not exist, then* $\omega \Longrightarrow_{P_E} \omega'$.

The reflexive and transitive closure of relation $\overset{\alpha}{\Longrightarrow}_\Gamma$ is denoted by $\overset{\alpha}{\Longrightarrow}^*_\Gamma$. If no confusion arises, then Γ can be omitted from the notation.

The language of a SEG system is the set of all environmental states that are reachable from the initial configuration by a sequence of direct derivation steps.

Definition 32 *The language generated by a* SEG *system* $\Gamma = (V, P_E, R_1, \ldots, R_n, \omega_E)$, $n \geq 1$*, in team derivation mode* α*, for* $\alpha \in \{d^{\diamondsuit q}, c^{\diamondsuit q}, t^{\triangle V_B} \mid \diamondsuit \in \{\leq, =, \geq\}, \triangle \in \{\subseteq, =, \supseteq\}, q \in \mathbb{N}_0, \emptyset \neq V_B \subseteq V\}$*, is defined by* $L(\Gamma, \alpha) = \{y \mid \omega_E \overset{\alpha}{\Longrightarrow}^*_\Gamma y\}$.

We illustrate how SEG systems work in various team modes through an example.

Example 5 *Let* $\Gamma = (\{a_1, a_2, \dots a_n\}, P_E, R_1, \dots, R_n, a_1a_2 \dots a_n)$, *where*

$$\begin{aligned} P_E &= \{a_1 \to a_1, a_2 \to a_2, \dots, a_n \to a_n\}, \text{ and} \\ R_i &= \{a_i \to a_i a_i\}, \text{ for } 1 \leq i \leq n. \end{aligned}$$

Notice that if $f_1 \in \{d^{=0}, d^{\leq 0}, d^{\geq 0}, d^{\leq q_1}, c^{=0}, c^{\leq 0}, c^{\geq 0}, c^{\leq q_1} \mid q_1 \geq 1\}$, *then* $L(\Gamma, f_1) = \{a_1^m a_2^m \dots a_n^m \mid m \geq 1\}$, *which is not a context–free language provided that* $n \geq 3$. *It can be seen that* $card(dom(R_i)) = 1$, $1 \leq i \leq n$. *Initially, the level of competence of an agent is* $lev(R_i, a_1a_2 \dots a_n) = 1$, *which does not alter during the derivation. It can be also verified that* $L(\Gamma, f_2) = \{a_1a_2 \dots a_n\}$ *and* $L(\Gamma, f_3) = \{a_1 \dots a_i^m \dots a_n \mid m \geq 1\}$, *where* $f_2 \in \{d^{=q_1}, d^{\geq q_1}, c^{=q_1}, c^{\geq q_1}, t^{\supseteq\{a_i\}}, t^{\triangle\{a_{j_1}, a_{j_2}, \dots a_{j_k}\}} \mid q_1 \geq 1, \triangle \in \{\subseteq, =, \supseteq\}, 1 \leq i \leq n, \{j_1, j_2, \dots j_k\} \subseteq \{1, 2, \dots n\}, 2 \leq k \leq n\}$, *and* $f_3 \in \{t^{=\{a_i\}}, t^{\subseteq\{a_i\}} \mid 1 \leq i \leq n\}$. *Note that both* $L(\Gamma, f_2)$ *and* $L(\Gamma, f_3)$ *are regular languages, moreover,* $L(\Gamma, f_2)$ *is finite.*

Let $\Gamma = (V, P_E, R_1, \dots, R_n, \omega_E)$, $n \geq 1$, be a SEG system as above. In the sequel, the class of languages generated by SEG systems with at most n agents using team derivation mode α is denoted by $\mathcal{L}(\text{SEG}(n, \alpha))$, where $\alpha \in \{d^{\diamondsuit q}, c^{\diamondsuit q} \mid \diamondsuit \in \{\leq, =, \geq\}, q \in \mathbb{N}_0\}$.

For an alphabet V and for some $\emptyset \neq V_B \subseteq V$, $\triangle \in \{\subseteq, =, \supseteq\}$, we denote by $\mathcal{L}(\text{SEG}(n, t^{\triangle V_B}))$ the class of languages produced by SEG systems with at most n agents employing team derivation mode $t^{\triangle V_B}$.

By definition we consider a 0L system as a SEG system with no agent.

Furthermore, we set $\mathcal{L}(\text{SEG}(\alpha)) = \bigcup_{n \geq 0} \mathcal{L}(\text{SEG}(n, \alpha))$, $\alpha \in \{d^{\diamondsuit q}, c^{\diamondsuit q} \mid \diamondsuit \in \{\leq, =, \geq\}, q \in \mathbb{N}_0\}$ and $\mathcal{L}(\text{SEG}(t^{\triangle V_B})) = \bigcup_{n \geq 0} \mathcal{L}(\text{SEG}(n, t^{\triangle V_B}))$ for some $\emptyset \neq V_B \subseteq V$, $\triangle \in \{\subseteq, =, \supseteq\}$.

5.3 Hierarchies and Relationships

In this section we investigate the language hierarchies induced by the number of agents, the computational power of SEG systems working in the team derivation modes above in comparison with other generative mechanisms such as certain Chomsky grammars and L systems.

5.3.1 Hierarchies

Our aim is to establish whether or not the language hierarchies induced by the number of agents are infinite.

Theorem 4 *Language hierarchies* $\mathcal{L}(\mathrm{SEG}(n-1, c^{\diamondsuit q})) \subseteq \mathcal{L}(\mathrm{SEG}(n, c^{\diamondsuit q}))$, $\diamondsuit \in \{\leq, =, \geq\}$, $q \geq 0$, *and* $\mathcal{L}(\mathrm{SEG}(n-1, \alpha)) \subseteq \mathcal{L}(\mathrm{SEG}(n, \alpha))$, *where* $n \geq 2$, $\alpha \in \{d^{=0}, d^{\geq 0}, d^{\leq q}, d^{\geq 1} \mid q \geq 0\}$, *are infinite.*

Proof
To prove the statement for team modes $d^{=0}, d^{\geq 0}, d^{\leq q_1}$, $c^{=0}, c^{\geq 0}$ and $c^{\leq q_1}$, $q_1 \geq 0$, consider the language $L = \{a_1^k \dots a_n^k \mid k \in \mathbb{N}\}$, which can be generated in all of the team modes above by the SEG system defined in Example 5.

L can also be produced in team mode $d^{\geq 1}$ by a SEG system similar to the SEG system presented in Example 5 except for the definition of agent R_i, which should be modified as follows: $R_i = \{a_j \to a_j^2 \mid 1 \leq j \leq i\}$, for all $1 \leq i \leq n$.

In order to support our claim, observe that $lev(R_i, \alpha) = i$ for each sentential form α, which signifies that the team consisting of all agents has to be employed in each derivation step. The only possible way of guaranteeing that all agents will be able to work is to choose rule $a_i \to a_i^2$ for agent R_i, $1 \leq i \leq n$.

To generate L in team modes $c^{=q}$, $c^{\geq q}$, $q \geq 1$, consider the system

$$\Gamma = (\{a_1, \dots, a_n, Y_1, \dots, Y_q\}, P_E, R_1, \dots, R_n, a_1 \dots a_n),$$

where $P_E = \{a_i \to a_i \mid 1 \leq i \leq n\} \cup \{Y_j \to Y_j \mid 1 \leq j \leq q\}$ and $R_i = \{a_i \to a_i^2\} \cup \{Y_j \to Y_j \mid 1 \leq j \leq q\}$, for all $1 \leq i \leq n$.

Assume now that L can be produced in any of the above mentioned team modes by a system with less than n agents, for $n \geq 2$. Since the number of each letter has to be increased by one in each derivation step, either some of the agents have to augment the number of more than one letters, or the number of at least one letter has to be multiplied by the environment. Strings that are not in L can be obtained in both cases, which leads to a contradiction. The detailed proof is left to the reader. ■

5.3.2 Finite Languages

We prove that finite languages satisfying certain conditions can be generated by SEG systems functioning in some of the above team modes.

Theorem 5 *Let V be an alphabet. Then, for any finite language $L = \{x_1, \ldots, x_n\}$, where $x_i \in V^*$, $1 \leq i \leq n$,*

- *in team derivation mode $d^{=q}$ and $d^{\geq q}$, $q \in \mathbb{N}$, it holds that $L \in \mathcal{L}(\mathrm{SEG}(d^{=q}))$ and $L \in \mathcal{L}(\mathrm{SEG}(d^{\geq q}))$, provided that $q + 1 \leq card(alph(x_i))$ for all i, $1 \leq i \leq n$;*
- *in team derivation modes $d^{\Diamond 0}$ and $d^{\leq q}$, $\Diamond \in \{\leq, =, \geq\}$, $q \in \mathbb{N}$, it can be verified that $L \in \mathcal{L}(\mathrm{SEG}(d^{\Diamond 0}))$ and $L \in \mathcal{L}(\mathrm{SEG}(d^{\leq q}))$;*
- *in team derivation modes $c^{\Diamond q}$, $q \in \mathbb{N}_0$, $\Diamond \in \{\leq, =, \geq\}$, it can be proved that $L \in \mathcal{L}(\mathrm{SEG}(c^{\Diamond q}))$;*
- *there is V_B, $\emptyset \neq V_B \subseteq V$, in team derivation mode $t^{\triangle V_B}$, $\triangle \in \{\subseteq, =, \supseteq\}$, such that the statement $L \in \mathcal{L}(\mathrm{SEG}(t^{\triangle V_B}))$ is valid.*

Proof
In the first place, it is easy to see that $L(\Gamma, t^{\triangle V_B}) = L(\Gamma, d^{\Diamond 0}) = L(\Gamma, d^{\leq q}) = L$, $\Diamond \in \{\leq, =, \geq\}$, $q \in \mathbb{N}$, $\emptyset \neq V_B = V$, $\triangle \in \{\subseteq, =, \supseteq\}$, where $\Gamma = (V, \{a \to \lambda \mid a \in V\}, \{a \to x_i \mid a \in V, 1 \leq i \leq n\}, x_1)$.

Secondly, it can be shown that $L(\Gamma, d^{=q}) = L(\Gamma, d^{\geq q}) = L$, $q \in \mathbb{N}$, where $q + 1 \leq card(alph(x_i))$ for all i, $1 \leq i \leq n$. Let us assume that $alph(L) = \{a_1, \ldots, a_{k+q}\}$, $k, q \in \mathbb{N}$. Indeed, the SEG system $\Gamma = (V, \{a \to \lambda \mid a \in V\}, (R_i)_{i=1}^{k+q}, (R^{\omega}_{\{j_1,\ldots,j_{q+1}\}})_{\{j_1,\ldots,j_{q+1}\} \in \mathcal{J}, \omega \in L}, x_1)$, $\mathcal{J} = \{\{j_1, \ldots, j_{q+1}\} \subseteq \{1, \ldots, k+q\} \mid k = card(alph(L)) - q\}$, where

$$R_i = \{a_i \to \lambda \mid a_i \in V\},$$

$$R^{\omega}_{\{j_1,\ldots,j_{q+1}\}} = \{a_{j_1} \to \delta, \ldots, a_{j_{q+1}} \to \delta \mid \{a_{j_1}, \ldots, a_{j_{q+1}}\} \subseteq V, \text{ where if } lev(R_{\{j_1,\ldots,j_{q+1}\}}, \omega) = q+1, \text{ then } \delta = x_s, 1 \leq s \leq n, \text{ or if } lev(R_{\{j_1,\ldots,j_{q+1}\}}, \omega) \neq q+1, \text{ then } \delta = \lambda\},$$

produces L in team derivation mode $d^{=q}$ and $d^{\geq q}$, $q \in \mathbb{N}$.

Finally, we will demonstrate that $L \in \mathcal{L}(\mathrm{SEG}(c^{\Diamond q}))$, $q \in \mathbb{N}$, $\Diamond \in \{\leq, =, \geq\}$. (Case $c^{=0}$ is proven in [33], a similar argument is valid for team modes $c^{\leq 0}$ and $c^{\geq 0}$). Suppose that $alph(L) = \{a_1, \ldots, a_r\}, r \geq 1$, and symbols $Y_1, \ldots, Y_q$ are not in $alph(L)$. Furthermore, let $\{x_{h_1}, \ldots, x_{h_s}\} \subseteq \{x_1, \ldots, x_n\} = L$, where $alph(x_{h_k}) \neq alph(x_{h_{k'}})$, if $1 \leq k \neq k' \leq s$. In fact, the SEG system $\Gamma = (V, \{a \to \lambda, Y_j \to \lambda \mid a \in V, 1 \leq j \leq q\}, (R^{\omega}_{h_z})^{\omega \in L}_{z=1,\ldots,s}, x_1)$, where

$$R^{\omega}_{h_z} = \{a \to \delta, Y_j \to x_s \mid a \in alph(x_{h_z}), 1 \leq j \leq q, x_s \in L, 1 \leq s \leq n, \text{ where if } alph(\omega) = alph(x_{h_z}), \text{ then } \delta = x_s, 1 \leq s \leq n, \text{ or if } alph(\omega) \neq alph(x_{h_z}), \text{ then } \delta = \lambda\},$$

generates L in team derivation modes $c^{\Diamond q}$, $q \in \mathbb{N}$, $\Diamond \in \{\leq, =, \geq\}$.

5.3.3 Regular and Context–Free Languages

Herein we show that the families of languages generated by SEG systems functioning in the team modes above are incomparable with the family of regular languages and context–free languages.

Theorem 6 *The following assertions can be verified:*

- *for each* $\vartheta \in \{d^{\Diamond q}, c^{\Diamond q} \mid \Diamond \in \{\leq, =, \geq\}, q \in \mathbb{N}_0\}$, *the family* $\mathcal{L}(\mathrm{SEG}(\vartheta))$ *is incomparable with* $\mathcal{L}(\mathrm{REG})$ *and* $\mathcal{L}(\mathrm{CF})$*;*
- *for each alphabet* V *with* $card(V) \geq 2$, *there exists* V_B, $\emptyset \neq V_B \subseteq V$, *such that the family* $\mathcal{L}(\mathrm{SEG}(t^{\triangle V_B}))$, $\triangle \in \{\subseteq, =, \supseteq\}$, *is incomparable with* $\mathcal{L}(\mathrm{REG})$ *and* $\mathcal{L}(\mathrm{CF})$.

Proof
We first note that $L = \{a^{2^n} \mid n \geq 0\}$ is in $\mathcal{L}(\mathrm{SEG}(\alpha))$ for any $\alpha \in \{c^{\Diamond 0}, c^{\leq q}, d^{\Diamond 0}, d^{\leq q} \mid \Diamond \in \{\leq, =, \geq\}, q \geq 1\}$, but this language is not context–free. Indeed, the SEG system $\Gamma = (\{a\}, \{a \to a^2\}, \{a \to a^2\}, a)$ generates L in any team mode α as above.

Secondly, we may observe that L is also in $\mathcal{L}(\mathrm{SEG}(\beta))$ for any $\beta \in \{c^{=q}, c^{\geq q} \mid q \geq 1\}$. In fact, it can be produced by the SEG system

$$\Gamma = (\{a, b_1, \ldots, b_q\}, P_E, R_1, a),$$

where $P_E = \{a \to a^2, b_i \to b_i \mid 1 \leq i \leq q\}$, $R_1 = \{a \to a^2, b_i \to b_i \mid 1 \leq i \leq q\}$ in any team mode β as above.

Thirdly, we may conclude that $L = \{a^{2^n} b_1 \dots b_q \mid n \geq 1\}$ is in $\mathcal{L}(\mathrm{SEG}(\gamma))$ for any $\gamma \in \{d^{=q}, d^{\geq q} \mid q \geq 1\}$, though this language is not context–free. In effect, the SEG system

$$\Gamma = (\{a, b_1, \dots, b_q\}, P_E, R_1, R_2, a^2 b_1 \dots b_q),$$

where $P_E = \{a \to a^2, b_i \to b_i \mid 1 \leq i \leq q\}$, $R_1 = \{a \to a^2\}$, and $R_2 = \{a \to a^2, b_i \to b_i \mid 1 \leq i \leq q\}$ generates L in any team mode γ as above.

Finally, notice that $L = \{a_1^{2^n} a_2^{2^n} \dots a_k^{2^n} \mid n \geq 0\}$ is in $\mathcal{L}(\mathrm{SEG}(\delta))$ for any $\delta = t^{\triangle V_B}$, $\triangle \in \{\subseteq, =, \supseteq\}$, $\emptyset \neq V_B \subseteq V$, since it can be produced by the SEG system

$$\Gamma = (\{a_1, \dots, a_k\}, \{a_i \to a_i^2 \mid 1 \leq i \leq k\}, \{a_i \to a_i^2 \mid 1 \leq i \leq k\}, a_1 a_2 \dots a_k)$$

in any of team mode $t^{\triangle V_B}$, $\triangle \in \{\subseteq, =, \supseteq\}$, for $V_B = \{a_1, \dots, a_k\}$.

On the other hand, the language $\{ab^n, ba^n \mid n \geq 1\}$ cannot be generated by any SEG system in any of the studied team modes ϑ, $\vartheta \in \{d^{\Diamond q}, c^{\Diamond q}, t^{\triangle V_B} \mid \Diamond \in \{\leq, =, \geq\}, \triangle \in \{\subseteq, =, \supseteq\}, q \in \mathbb{N}_0, \emptyset \neq V_B \subseteq V\}$. To see this, note that team formation is based on the letter occurrence of the strings, therefore SEG systems of the types above cannot produce strings of forms ab^i and ba^i, $i \geq 1$, without generating words of different forms. The detailed proof is left to the reader. ∎

5.3.4 L Systems

It follows directly from the definitions that the family of 0L languages is included in any of the families $\mathcal{L}(\mathrm{SEG}(\alpha))$, where $\alpha \in \{c^{\Diamond q}, d^{\Diamond q}, t^{\triangle V_B} \mid \Diamond \in \{\leq, =, \geq\}, \triangle \in \{\subseteq, =, \supseteq\}, \emptyset \neq V_B \subseteq V, q \in \mathbb{N}_0\}$. According to Example 5 the inclusion is strict for $\alpha \in \{d^{=0}, d^{\geq 0}, d^{\leq q}, c^{=0}, c^{\geq 0}, c^{\leq q} \mid q \in \mathbb{N}_0\}$.

Theorem 7 *The claims below are valid:*

- *for each* $\alpha \in \{d^{\Diamond q}, c^{\Diamond q} \mid \Diamond \in \{\leq, =, \geq\}, q \in \mathbb{N}_0\}$, $\mathcal{L}(\mathrm{SEG}(\alpha))$ *and* $\mathcal{L}(\mathrm{T0L})$ *are incomparable;*

- *for each alphabet V, there exists V_B, $\emptyset \neq V_B \subseteq V$, such that $\mathcal{L}(\mathrm{SEG}(t^{\triangle V_B}))$, $\triangle \in \{\subseteq, =, \supseteq\}$, and $\mathcal{L}(\mathrm{T0L})$ are incomparable.*

Proof
The proof for team derivation modes $c^{=0}, d^{=0}$ and $t^{=V_B}$ for $\emptyset \neq V_B = V$ can be found in [33]. An analogous argument is valid for team modes $d^{\geq 0}, d^{\leq q}$, $c^{\geq 0}, c^{\leq q}$, $t^{\triangle\{a\}}$, $q \in \mathbb{N}_0$, $\triangle \in \{\subseteq, =, \supseteq\}$, and the non–T0L language $L_1 = \{a, a^3\}$, which can be generated by the SEG system $\Gamma_1 = (\{a\}, \{a \to \lambda\}, \{a \to a^3\}, a)$.

A similar system with alphabet $V = \{a, b_1, \ldots, b_q\}$ and agent $\{a \to a^3, b_i \to b_i \mid 1 \leq i \leq q\}$ produces L_1 in team modes $c^{=q}$ and $c^{\geq q}$.

The non–T0L language $L_2 = \{a_1 a_2 \ldots a_{2+q}, a_1^3 a_2^3 \ldots a_{2+q}^3\}$ can be generated in team modes $d^{=q}$ and $d^{\geq q}$ by the system

$$\Gamma_2 = (\{a_1, \ldots, a_{q+2}\}, \{a_i \to \lambda \mid 1 \leq i \leq q+2\}, R_1, R_2, a_1 \ldots a_{q+2}),$$

where $R_1 = \{a_1 \to a_1^3\}$ and $R_2 = \{a_i \to a_2^3 \cdots a_{q+2}^3 \mid 2 \leq i \leq q+2\}$.

The verification of the fact that the T0L language $L_3 = \{a^{2^n 3^m} \mid n, m \geq 1\} \notin \mathcal{L}(\mathrm{SEG}(\alpha))$ for any $\alpha \in \{c^{=0}, d^{=0}, t^{=\{a\}}\}$, can be found in [33]. Analogous considerations may be applied to $\alpha \in \{c^{\leq q}, c^{\geq q}, c^{=q}, d^{\leq q}, t^{\supseteq\{a\}}, t^{\subseteq\{a\}} \mid q \in \mathbb{N}_0\}$. Furthermore, it is clear that L_3 cannot be generated in team modes $d^{=q}$ and $d^{\geq q}$, $q \in \mathbb{N}$, since $alph(L_3) = \{a\}$, whereas the alphabet of a SEG system working in these team modes has to contain at least $q+1$ elements. ∎

5.4 The Power of Team Cooperation

In the sequel, we will demonstrate that team cooperation leads to quite a large computational power. Herein we will not deal with team modes $c^{=0}, t^{=V}$, $\emptyset \neq V$, since the verification of the theorems below for these cases can be found in [33]. The SEG systems employed in [33] can also be used to support the claim for team modes $c^{\leq 0}, c^{\geq 0}, t^{\subseteq V}, t^{\supseteq V}$, $\emptyset \neq V$.

First, let us consider team derivation modes $c^{=q}$ and $c^{\leq q}$, $q \in \mathbb{N}$.

Theorem 8 *A language L over an alphabet T is recursively enumerable, if and only if it can be obtained as $L = L' \cap T^*$ for some $L' \in \mathcal{L}(\mathrm{SEG}(\alpha))$, where $\alpha \in \{c^{=q}, c^{\leq q} \mid q \in \mathbb{N}\}$.*

Proof

Actually, we will show that every language generated by a context–free random context grammar can be expressed in the form claimed by the theorem. Since any recursively enumerable language can be produced by a context–free random context grammar [36, 110], the assertion follows.

Without loss of generality, it may be assumed that L is generated by a context–free random context grammar $G = (N, T, S, P)$, whose productions are of the form $(B, C) : A \to x$, where $A \to x$ is a context–free production and B, C are two nonterminals. Suppose that the productions in P are labelled in a one–to–one manner by numbers from 1 to n and let $N = \{A_1, \ldots, A_r\}$, $r \geq 1$.

We will construct a SEG system working in team mode $c^{=q}$, $q \in \mathbb{N}$, which simulates the derivations in G.

Let $X, X_i, X_i', X_i'', Y_j, X_D, X_D'$ and F be new distinct symbols not in $N \cup T$, where $1 \leq i \leq n,\ 1 \leq j \leq q,\ D, D' \in N$. We define

$$\Gamma = (V, P_E, R_0, (R_i, R_i', R_i'')_{1 \leq i \leq n}, (R_D)_{D \in N}, (R_D')_{D' \in N}, XS),$$

where

$$\begin{aligned}
V &= N \cup T \cup \{X, X_i, X_i', X_i'', Y_j, F \mid 1 \leq i \leq n,\ 1 \leq j \leq q\} \cup \{X_D, X_D' \mid D, D' \in N\},\\
P_E &= \{a \to a \mid a \in V \setminus \{\{X_i'' \mid 1 \leq i \leq n\} \cup \{X_D, X_D' \mid D, D' \in N\}\}\} \cup\\
&\quad \{X_D \to F \mid D \in N\} \cup \{X_D' \to F \mid D' \in N\} \cup \{X_i'' \to X \mid 1 \leq i \leq n\},\\
R_0 &= \{X \to X_i X_C, X \to \lambda, Y_j \to F \mid 1 \leq i \leq n,\ 1 \leq j \leq q\},
\end{aligned}$$

for each $D \in N$,

$$R_D = \{X_D \to \lambda, D \to D, Y_j \to F \mid 1 \leq j \leq q-1\},$$

for each $D' \in N$,

$$\begin{aligned}
R_D' = \{&X_D' \to \chi, D' \to \psi, Y_j \to F \mid 1 \leq j \leq q, \text{ where if } D' \neq C, \text{ then } \chi = \lambda, \psi = D',\\
&\text{or if } D' = C, \text{ then } \chi = \psi = F\},
\end{aligned}$$

and for each rule $(B, C) : A \to x$ labelled by $i,\ 1 \leq i \leq n$, we introduce agents

$$\begin{aligned} R_i &= \{X_i \to X_i', Y_j \to F \mid 1 \leq j \leq q\}, \\ R_i' &= \{X_i' \to X_i'' X_{A_1} \dots X_{A_r}, \delta \to F, Y_j \to F \mid 1 \leq j \leq q, \text{ where if } B \neq \emptyset, \text{ then } \delta = B, \\ &\quad \text{or if } B = \emptyset, \text{ then } \delta = A\}, \\ R_i'' &= \{X_i'' \to F, A \to x, Y_j \to F \mid 1 \leq j \leq q\}. \end{aligned}$$

We explain how a derivation in G can be simulated by some derivation in Γ in team mode $c^{=q}$, $q \in \mathbb{N}$. Let us assume that at some moment the current state of Γ is $X\omega$, where ω is a sentential form in G. Note that the initial state of Γ is a string of this form. A derivation step in G, where rule $(B, C) : A \to x$ labelled by i is to be applied may be simulated as follows:

1. During the first stage, only agent R_0 and agents R_D can work, where $D \in N$ and D appears in ω. R_0 indicates the rule that has to be simulated by rewriting X to $X_i X_C$, $1 \leq i \leq n,\ C \in N$, where C is the forbidding context condition. The work of agents R_D does not change the sentential form.

2. In the second phase, the simulation is continued by a team formed from agent R_i, agents R_D and agents $R_{D'}$, where $D \in N$, $D \neq C$ occurs in the current sentential form and either $D = C$, D is absent from or $D' \in N$, $D' = C$ is present in the sentential form. R_i rewrites X_i to X_i' and X_C is either erased by agent R_D, if $D \in N$, $D = C$ does not appear in the sentential form or it is substituted for the trap symbol F as a result of the action of agent $R_{D'}$, if $D' \in N$, $D' = C$ is present in the sentential form. The trap symbol F cannot be removed from the string. If $D \in N, D \neq C$, then agent R_D performs an identical rewriting on D. No other agents can be active during this derivation phase.

3. At the third stage, a team formed from agent R_i' and agents R_D, where $D \in N$ and D appears in the sentential form is active. R_i' rewrites X_i' to $X_i'' X_{A_1} \dots X_{A_r}$, if there is a symbol B (or symbol A, if $B = \emptyset$) in the sentential form. The appearance of the symbol X_i'' in the sentential form indicates that the conditions of application of the production labelled by i are satisfied. Symbols X_{A_j}, $1 \leq j \leq r$, guarantee the

correctness of the next step of the simulation. The work of agents R_D, $D \in N$, does not alter the sentential form, i.e. identical rewritings are applied. No other agents can be active in this phase of the derivation.

4. In the fourth phase, agent R_i'' applies the production labelled by i to substitute A with x. Furthermore, for any $D' \in N$ present in and any $D \in N$ absent from the sentential form agents R_D' and R_D should be activated, as well. Only in the case when $D' \in N$, $D' \neq C$ does the derivation terminate without the introduction of the trap symbol: R_D' and R_D rewrite $X_{D'}$ and X_D, respectively, to λ, the environment replaces X_i'' with X and performs some identical rewritings, which completes the simulation of the production.

Should agent R_0 erase the nonterminal X in lieu of initiating the simulation of a production, then the derivation cannot produce any new strings. In such a case, if the sentential form is a terminal string, then it is an element of $L(G)$. Due to the construction of the SEG system Γ, only words of $L(G)$ can be obtained. Thus our statement is verified for team mode $c^{=q}$, $q \in \mathbb{N}$.

To prove our claim for team mode $c^{\leq q}$, $q \in \mathbb{N}$, symbols D', X_D' and agents $R_{D'}$ should be omitted from the definition of Γ. Agent R_D, $D \in N$, ought to be modified as follows:

$$\begin{aligned} R_D \;=\; & \{X_D \to \chi, D \to \psi, Y_j \to F \mid 1 \leq j \leq q-1, \text{ where if } D \neq C, \text{ then } \chi = \lambda, \psi = D, \\ & \text{or if } D = C, \text{ then } \chi = \lambda, \psi = F\}. \end{aligned}$$

The verification is analogous to that of team mode $c^{=q}$, $q \in \mathbb{N}$, consequently, it is left to the reader. ∎

Secondly, we focus on team derivation mode $d^{=q}$, $q \in \mathbb{N}$.

Theorem 9 *A language L over an alphabet T is recursively enumerable, if and only if it can be obtained as $L = L' \cap T^*$ for some $L' \in \mathcal{L}(\mathrm{SEG}(d^{=q}))$, $q \in \mathbb{N}$.*

Proof

Let us consider the context–free random context grammar of the same form as in the case of the previous proof. We will construct a SEG system working in team mode $d^{=q}$, $q \in \mathbb{N}$, which simulates the derivations in G.

Let $X, F, X_k, X'_k, X''_k, X_{n+1}, Y_l, Y_{k,j}, Y'_{k,j}, Y''_{k,j}, Y_{n+1,j}, 1 \leq k \leq n, 1 \leq l \leq q+1, 1 \leq j \leq q+4$, be new distinct symbols not in $N \cup T$. We define

$$\Gamma = (V, P_E, R_0^1, R_0^2, (R_k^i)_{1 \leq k \leq n, 1 \leq i \leq 6}, R_{n+1}^1, R_{n+1}^2, XY_1 \dots Y_{q+1}S),$$

where $V = N \cup T \cup \{F, X\} \cup \{X_k, X'_k, X''_k, Y_{k,j}, Y'_{k,j}, Y''_{k,j} \mid 1 \leq k \leq n, 1 \leq j \leq q+4\} \cup \{Y_l \mid 1 \leq l \leq q+1\} \cup \{X_{n+1}\} \cup \{Y_{n+1,j} \mid 1 \leq j \leq q+4\}$,

$$\begin{aligned} P_E = {} & \{a \to a \mid a \in N \cup T \cup \{F, X, X_k, X''_k \mid 1 \leq k \leq n\}\} \cup \{X'_k \to X''_k \mid 1 \leq k \leq n\} \cup \\ & \{X_{n+1} \to \lambda\} \cup \{Y_{k,j} \to Y'_{k,j}, Y'_{k,j} \to Y''_{k,j}, Y''_{k,j} \to \lambda \mid 1 \leq k \leq n, 1 \leq j \leq q+4\} \cup \\ & \{Y_j \to \lambda \mid 1 \leq j \leq q+1\} \cup \{Y_{n+1,j} \to \lambda \mid 1 \leq j \leq q+4\}. \end{aligned}$$

We also have

$$\begin{aligned} R_0^1 &= \{X \to X_k Y_{k,1} \dots Y_{k,q+4}, Y_{q+1} \to F \mid 1 \leq k \leq n+1\}, \\ R_0^2 &= \{X \to F, Y_j \to \lambda \mid 1 \leq j \leq q+1\}, \\ R_{n+1}^1 &= \{X_{n+1} \to \lambda, Y_{n+1,j} \to F \mid q+1 \leq j \leq q+4\}, \\ R_{n+1}^2 &= \{X_{n+1} \to F, Y_{n+1,j} \to \lambda \mid 1 \leq j \leq q+4\}, \end{aligned}$$

and for each rule $(B, C) : A \to x$, labelled by k, $1 \leq k \leq n$, we introduce agents

$$\begin{aligned} R_k^1 &= \{X_k \to X'_k\}, \\ R_k^2 &= \{X_k \to F, \delta \to F, Y_{k,j} \to Y'_{k,j} \mid 1 \leq j \leq q-1 \text{ , where if } B \neq \emptyset, \\ &\qquad \text{then } \delta = B, \text{ or if } B = \emptyset, \text{ then } \delta = A\}, \\ \\ R_k^3 &= \{C \to F, Y'_{k,q+4} \to F\}, \\ R_k^4 &= \{X'_k \to X''_k, Y'_{k,j} \to Y''_{k,j} \mid 1 \leq j \leq q+1\}, \\ \\ R_k^5 &= \{X''_k \to F, A \to x, Y_{k,q+4} \to F, Y''_{k,q+4} \to F\}, \\ R_k^6 &= \{X''_k \to XY_1 \dots Y_{q+1}, Y''_{k,j} \to F \mid 1 \leq j \leq q+2\}. \end{aligned}$$

We clarify how a derivation in G can be simulated by some derivation in Γ in team derivation mode $d^{=q}$, $q \in \mathbb{N}$. Assume that at some moment the current state of Γ is $XY_1 \dots Y_{q+1}\omega$, where ω is a sentential form in G. Notice that the initial state of Γ is a string of this form. A derivation step in G, where rule $(B, C) : A \to x$ labelled by k is to be employed may be simulated as follows:

1. During the initial phase, only team $\mathcal{T}_0 = \{R_0^1, R_0^2\}$ can work. Agent R_0^1 indicates the rule to be simulated by rewriting X to $X_k Y_{k,1} \dots Y_{k,q+4}$, $1 \leq k \leq n$. This component may also terminate the simulation process through the substitution of X for $X_{n+1}Y_{n+1,1} \dots Y_{n+1,q+4}$. Component R_0^2 acts in parallel with R_0^1. R_0^2 erases one of the markers Y_j, $1 \leq j \leq q+1$. The other Y_ls, $1 \leq l \leq q+1, l \neq j$, will be deleted by the environment.

2. At the second stage, the simulation is continued by team $\mathcal{T}_1 = \{R_k^1, R_k^2\}$. This team can be formed and used, if and only if symbol B (or symbol A, if $B = \emptyset$) occurs in the sentential form. If symbol B (or symbol A, if $B = \emptyset$) is present in the sentential form, R_k^1 replaces X_k' with X_k'' and R_k^2 substitutes one of the markers, $Y_{k,j}$ for $Y_{k,j}'$. The other symbols will be rewritten by the environment.

3. During the third phase, the team that can be activated is $\mathcal{T}_2 = \{R_k^3, R_k^4\}$, which checks the absence of C in the sentential form. Should C appear in the sentential form, then the trap symbol will be introduced, which cannot be removed from the string. If C is not present in the sentential form, then the environment continues the derivation.

4. Finally, team $\mathcal{T}_3 = \{R_k^5, R_k^6\}$ will execute rule $A \to x$ and introduce nonterminals $X, Y_1, \dots, Y_{q+1}$, whereas the other markers are removed by the developmental rules of the environment.

When team $\mathcal{T}_4 = \{R_{n+1}^1, R_{n+1}^2\}$ is activated, all markers are removed, therefore none of the agents of Γ can be applied anymore. Owing to the construction of the SEG system Γ, only words of $L(G)$ can be produced. Thus our statement is proved for team mode $d^{=q}$, $q \in \mathbb{N}$.■

Lastly, we deal with team derivation modes $t^{=V_B}$ and $t^{\supseteq V_B}$, $\emptyset \neq V_B \subseteq V$.

Theorem 10 *For any recursively enumerable language $L \subseteq T^*$, there exist a* SEG *system* $\Gamma = (V, P_E, R_1, \ldots, R_m, \omega_E)$, $m \geq 1$, *and* V_B, $\emptyset \neq V_B \subseteq V$, *such that* $L = L' \cap T^*$ *holds, where* $L' \in \mathcal{L}(\mathrm{SEG}(\alpha))$, $\alpha \in \{t^{=V_B}, t^{\supseteq V_B}\}$.

Proof

Let us consider the context–free random context grammar G of the same form as in the previous proofs. Let us construct a SEG system, which simulates a derivation in G. Let X, F, X_i, X_i', X_i'', $1 \leq i \leq n$, X_{n+1}, be new distinct symbols not in $N \cup T$. The underlying SEG system is as follows:

$$\Gamma \quad = \quad (V, P_E, R_0, R_{n+1}, (R_{i,j})_{1 \leq i \leq n, 1 \leq j \leq 5}, XS),$$

where

$$\begin{aligned} V &= N \cup T \cup \{X, F\} \cup \{X_i, X_i', X_i'' \mid 1 \leq i \leq n\} \cup \{X_{n+1}\}, \\ V_B &= N \cup \{X, X_i, X_i', X_i'' \mid 1 \leq i \leq n\} \cup \{X_{n+1}\}, \\ P_E &= \{a \to a \mid a \in V\}, \end{aligned}$$

and

$$\begin{aligned} R_0 &= \{X \to X_i \mid 1 \leq i \leq n+1\} \cup \{D \to D \mid D \in N\}, \\ R_{n+1} &= \{X_{n+1} \to \lambda\}. \end{aligned}$$

Moreover, for each rule $(B, C) : A \to x$ labelled by i, $1 \leq i \leq n$, we introduce agents

$$\begin{aligned} R_{i,1} &= \{\delta \to F, X_i \to X_i', D \to F \mid D \in N \setminus \{\delta\}, \text{ where if } B \neq \emptyset, \\ &\qquad \text{then } \delta = B, \text{ or if } B = \emptyset, \text{ then } \delta = A\}, \\ R_{i,2} &= \{\delta \to \delta, X_i \to F, D \to F \mid D \in N \setminus \{\delta\}, \text{ where if } B \neq \emptyset, \\ &\qquad \text{then } \delta = B, \text{ or if } B = \emptyset, \text{ then } \delta = A\}, \\ R_{i,3} &= \{X_i' \to X_i'', D \to F \mid D \in N \setminus \{C\}\}, \\ R_{i,4} &= \{X_i'' \to X, D \to F \mid D \in N \cup \{X_i'\}\}, \\ R_{i,5} &= \{X_i'' \to F, A \to x, D \to F \mid D \in (N \cup \{X_i'\}) \setminus \{A\}\}. \end{aligned}$$

We demonstrate how an arbitrary derivation in G can be simulated by a derivation in Γ applying analogous considerations to the ones of the proof of the previous two statements. Suppose that at some moment the current state of Γ is $X\omega$, where ω is a sentential form in G. Indeed, the initial state of Γ is of this form. A derivation step in G during which rule $(B, C) : A \to x$ labelled by i is employed, can be simulated as follows:

1. At the first step, only component R_0 can work. It indicates the rule to be simulated: it rewrites X to X_i, $1 \leq i \leq n$. Observe that this component may also terminate the simulation process through the substitution of X with X_{n+1}.

2. At the second stage, the team consisting of $R_{i,1}$ and $R_{i,2}$ continue the derivation. The simulation works, if and only if $R_{i,1}$ uses rule $X_i \to X_i'$ and $R_{i,2}$ rule $B \to B$ (or $A \to A$, if $B = \emptyset$). In this way we can check whether the permitting context condition is satisfied.

3. During the third phase, we examine the fulfillment of the forbidding context condition. In the third phase, only team consisting of agent $R_{i,3}$ alone is allowed to be activated provided that the sentential form does not contain any C.

4. If the forbidding context condition is satisfied, then at the fourth stage team formed by $R_{i,4}$ and $R_{i,5}$ is activated and it finishes the simulation of production i through the replacement of an occurrence of A with x and X_i'' with X.

If symbol X is rewritten to X_{n+1}, R_{n+1} erases X_{n+1} from the sentential form. None of the agents can be employed anymore, after R_{n+1} has been activated, since all markers have been removed from the environmental string. Due to the construction of the SEG system Γ, only words of $L(G)$ can be generated. Consequently, our assertion is verified for team derivation modes $t^{=V_B}$ and $t^{\supseteq V_B}$, $\emptyset \neq V_B \subseteq V$. ∎

5.5 Discussion

In this chapter we have studied the bottom–up clustering algorithm [9, 20, 38, 81] in terms of simple eco–grammar systems [31]. We have extended the conditions of dynamic

team constitution proposed in [33]. We have established the relationships of simple eco–grammar systems formed according to the newly introduced conditions to each other. We can conclude that in certain team derivation modes the number of agents plays a crucial part in cluster formation. It has been investigated how simple eco–grammar systems consisting of dynamically formed teams given various constitution modes are related to the language classes of the Chomsky hierarchy. From the language classes these systems are capable of generating, we can deduce the difficulty of the problem they can solve. We have had to impose various restrictions on the alphabet of finite languages so that we are able to generate them by simple eco–grammar systems. We have shown that the cooperation of teams leads to quite a large computational power.

5.5.1 Related Work and Outlook

In the sequel, we compare our approach with the approaches of contemporary research and propose some further research directions.

First, bottom–up clustering is a way of extracting community structure from the network [96]. The extraction is based on the measure of the connection strength. As a result of the clustering process, the network will be divided into densely connected subgraphs, i.e. clusters. The communication between nodes belonging to different subgraphs is minimized. In our work, we have regarded the connection strength as the difference in the levels of competence/excitation of the agents. The communication of the agents can be viewed as the emergence of their intensive interaction with their commonly shared environment [30]. The communication between two teams is minimized, since at a given instant only the agents of the same team are allowed to alter the environmental string. Our approach differs in some aspects from the approaches presented in the literature [25, 96]. On the one hand, in our work, the communication occurs in an indirect manner. The agents perform computations and include the results in the environmental string. On the other hand, in the bottom–up clustering algorithm teams can work in a parallel manner [25, 96], which implies that the concept of teams of a team (subclusters of a cluster [25]) may be elaborated in our mathematical model, hence the potential of construction of a hierarchical structure.

Secondly, the amelioration of the performance of the bottom–up clustering algorithm is investigated from two different points of view in the literature [9, 20, 38, 59, 79]. It is questionable whether the modifications incorporated into the bottom–up clustering algorithm produce better answers than previous solutions (according to well–defined quality measure) or the same answers, in less time (in theory or in practice). In our model the goodness of the answer corresponds to the largeness of the language class that can be obtained due to the collaboration of the agents in various team modes. The number of time steps can be measured in terms of derivation steps, if we assume that one time step is equal to one team derivation step.

Finally, other team derivation modes can be introduced. The difference in the levels of competence/excitation of the agents may be modified over time. Hybrid team derivation modes [18, 45, 92, 107] can be considered. The members of different teams may also use different derivation modes. If we apply the external hybridization, then dynamic teams perform the derivation according to different conditions at various derivation steps. Whereas if we employ the internal hybridization, then the different conditions of dynamic team constitution conditions are combined by means of logical operators. The restrictions imposed on the levels of competence/excitation of the agents interpret the requirements of participation in the problem solving process. The properties of simple eco–grammar systems with dynamically formed teams of agents having different levels of competence/excitation over time or working either according to the external or the internal hybridization are the subjects of further research.

Summary

In this dissertation we have provided a formal language theoretic approach to self–organizing networks in terms of grammar systems. We have modelled these networks at syntactical level as well as we have argued some semantical and experimental aspects related to them.

First, we have applied networks of parallel multiset string processors with teams of collective and individual filtering to model peer–to–peer systems. We have established the connection between the growth of the number of strings being present during the computation at the components of these networks and the growth functions of certain types of developmental systems (Lindenmayer systems). The formal language theoretic description of P2P networks has proven to be advantageous, since it renders them adaptive and makes task–based dynamic configuration as well as the execution of pipelined operations possible owing to the introduction of the notion of the apprentice peer. We have demonstrated how the formal language theoretic model can be employed to incorporate network security requirements. More specifically, we have shown how to model and detect SYN flooding attacks and enforce Discretionary Access Control. Our approach allows quick and efficient local analysis of security requirements: testing is restricted to testing of a peer and peers engage in communication with the given peer, thus combinatorial tests can be avoided.

Secondly, we have described the search strategy of Internet crawlers in quest of novel information on different topics on the World Wide Web. We have illustrated the great diversity of employing regulated rewriting devices in simple eco–grammar systems. We have verified that if we ignore the aging of the World Wide Web, then these systems determine the class of recursively enumerable languages. Whereas if the web pages may become

obsolete, then the language family generated by unordered scattered context grammars of finite index can be obtained. We have used simulations to study the behaviour of our model crawlers in different graph topologies. We have compared the selective learning algorithm to the linear function approximation–based reinforcement learning algorithm. We have observed that in the topic–specific case the relative performance of the combined learning algorithm has improved in scale–free small worlds, in scale–free worlds and in random world environment. If the task has become more complex and the work sharing has been enforced by the environment, then the combined learning algorithm is at least equal, even superior to both the selective and the reinforcement learning algorithms in most cases. Furthermore, the communication has ameliorated the performance by a large margin and adaptive communication has proven to be advantageous in the majority of the cases.

Finally, to model the behaviour of agents organized into clusters, we have imposed various conditions on the determination of the simultaneously active groups of agents in simple eco–grammar systems. From the language classes that these systems are capable of generating, we may deduce the difficulty of the problem the agents can solve. We have investigated how these simple eco–grammar systems given the various team constitution modes are related to the language classes of the Chomsky hierarchy and developmental systems and whether they are able to produce any recursively enumerable language.

To sum up, for all self–organizing networks presented in this dissertation, we have also proposed some further research directions.

Összefoglalás

A disszertációban az önszervező hálózatok modellezésével foglalkoztunk. Szintaktikus leírásuk mellett bizonyos jellemzőiket szemantikus és kísérleti szempontból is megvizsgáltuk.

Először a peer–to–peer rendszerek jellemzésére multihalmazokat feldolgozó, csoportokba szerveződő, csoportszintű (kollektív) és nyelvprocesszor szintű (egyéni) szűrőket használó, párhuzamos nyelvprocesszor hálózatokat alkalmaztunk. Megállapítottuk, hogy kapcsolat áll fenn ezen párhuzamos nyelvprocesszor hálózatok komponenseinél a számítási lépések során megjelenő szavak számának növekedése és bizonyos típusú fejlődő rendszerek (Lindenmayer rendszerek) növekedési függvényei között. A P2P rendszerek formális nyelvi megfogalmazása több szempontból is előnyösnek bizonyult: a tanonc peerek (apprentice peers) fogalmának bevezetése lehetővé tette az adaptívitás megfogalmazását, valamint a feladatalapú dinamikus konfigurálás és a pipeline számítások végrehajtását a P2P hálózatokban. Azt is megmutattuk, hogyan építhetők be biztonsági követelmények a matematikai modellbe: hogyan modellezhető és detektálható a SYN csomagokkal történő elárasztásos támadás (SYN flooding attack) és hogyan valósítható meg a Discretionary Access Control. Az általunk alkalmazott megközelítés lehetővé teszi a biztonsági követelmények lokális ellenőrzését: a tesztelés az adott peerre és a peerrel kommunikációt folytató peerekre vonatkozik, így a kombinatorikus tesztelés elkerülhető.

Ezután a hálózati barangolók különböző témákban történő információkeresését fogalmaztuk meg formális nyelvi eszközökkel. A reguláris átírási eszközök széles körű alkalmazhatóságát illusztráltuk az egyszerű öko–grammatikarendszerekben. Igazoltuk, hogy ha eltekintünk a weblapok elévülésétől, akkor ezen egyszerű öko–grammatikarendszerek képesek a rekurzívan felsorolható nyelvek családjának előállítására. Ha viszont figyelembe vesszük a webla-

pok elévülését, akkor csak minden véges indexű, rendezetlen szétszórt szövegfeltételekkel adott grammatika (unordered scattered context grammar) által generált nyelvet lehet a segítségükkel meghatározni. A barangolók viselkedését számítógépes szimulációk alapján is megvizsgáltuk különböző gráftopológiákban. Összehasonlítottuk a szelektív tanulási algoritmust a lineáris függvényapproximációs alapú megerősítéses tanulással. Megállapítottuk, hogy a kombinált tanulási algoritmust alkalmazó témaspecifikus barangolóknál az algoritmus relatív teljesítménye skálamentes kisvilágban, skálamentes világban és véletlen gráfok esetében is javult. Amikor a feladat összetettebbé vált és a környezet a munkamegosztást szükségessé tette, akkor a kombinált tanulási algoritmus teljesítménye a legtöbb esetben legalább annyira jónak, ha nem jobbnak bizonyult, mint akár a szelektív tanulásé, akár a megerősítéses tanulásé. Továbbá a kommunikáció a teljesítményt is növelte és az adaptáció az esetek többségében előnyös volt.

Végül a klaszterekbe szerveződő ágensek viselkedésének leírására olyan egyszerű öko–grammatikarendszereket vezettünk be, amelyek az egyidejűleg aktív ágensek csoportjának meghatározását különböző feltételekhez kötik. A nyelvosztályokból, amelyeket az egyszerű öko–grammatikarendszerek generálnak, a rendszerek által megoldható feladatok nehézségére következtethetünk. Megvizsgáltuk az egyszerű öko–grammatikarendszerek által generált nyelvek kapcsolatát a Chomsky–hierarchia nyelvosztályaival és a Lindenmayer rendszerek által generált nyelvosztályokkal, valamint azt is, hogy vannak–e a különböző csoportképzési feltételek alapján szerveződő egyszerű öko–grammatikarendszerek között olyanok, amelyek képesek a rekurzívan felsorolható nyelvek előállítására?

A disszertációban bemutatott önszervező hálózatok esetében további kutatási lehetőségeket is javasoltunk.

Bibliography

[1] Peer–to–Peer, `http://en.wikipedia.org/wiki/Peer-to-Peer`, 2010.

[2] Adams, W. J., Davis, N. J.: Toward a Decentralized Trust–based Access Control System for Dynamic Collaboration, in: *Proceedings of the 6th Annual IEEE Workshop on Information Assurance*, United States Military Academy, West Point, NY, June 2005, 317–324.

[3] Albert, R., Barabási, A. L.: Statistical Mechanics of Complex Networks, *Reviews of Modern Physics*, **74**, 2002, 47–91.

[4] Albert, R., Jeong, H., Barabási, A. L.: The Internet's Achilles' Heel: Error and Attack Tolerance in Complex Networks, *Nature*, **406**, 2000, 378382.

[5] Arthi, K., Krithivasan, K.: Probabilistic Parallel Communicating Grammar Systems, *International Journal of Computer Mathematics*, **79**(1), 2002, 1–26.

[6] Arthi, K., Krithivasan, K., Csuhaj-Varjú, E.: On the Number of Rules in Components of Cooperating Grammar Systems with Probabilities, *Journal of Automata, Languages and Combinatorics*, **7**, 2002, 433–446.

[7] Ashby, W. R.: Principles of the Self–Organizing Dynamic System, *Journal of General Psychology*, **37**, 1947, 125–128.

[8] Ashby, W. R.: Principles of the Self–Organizing System, in: *Principles of Self–Organization: Transactions of the University of Illinois Symposium* (H. von Foerester, G. W. Zopf Jr, Eds.), Pergamon Press, London, UK, 1962, 255–278.

[9] Baeza-Yates, R., Ribeiro-Neto, B.: *Modern Information Retrieval*, Addision Wesley, Boston, MA, USA, 1999.

[10] Bak, P.: *How Nature Works: The Science of Self–Organized Criticality*, Copernicus, New York, NY, USA, 1996.

[11] Barabási, A.: The Architecture of Complexity: From Network Structure to Human Dynamics, *IEEE Control Systems Magazine*, 2007, 33–42.

[12] Barabási, A., Albert, R.: Emergence of Scaling in Random Networks, *Science*, **286**, 1999, 509–512.

[13] Barabási, A., Bonabeau, E.: Scale–Free Networks, *Scientific American*, 2003, 50–59.

[14] Barabási, A. L.: *Linked: The New Science of Networks*, Perseus Publishing, Cambridge, MA, USA, 2002.

[15] Barabási, A. L., Albert, R., Jeong, H.: Scale–Free Characteristics of Random Networks: the Topology of the World–Wide Web, *Physica A*, **281**, 2000, 69–77.

[16] Bollobás, B.: *Random Graphs*, Academic, London, UK, 1985.

[17] Bonatti, P. A., Duma, C., Olmedilla, D., Shahmehri, N.: An Integration of Reputation–based and Policy–based Trust Management, in: *Proceedings of the Semantic Web and Policy Workshop*, Galway, Ireland, November 2005, 136–141.

[18] Bordihn, H., Holzer, M.: Grammar Systems with Negated Conditions in Their Cooperation Protocols, *Journal of Universal Computer Science*, **6**(12), 2000, 1165–1184.

[19] Bornholdt, S., Ebel, H.: World Wide Web Scaling Exponent from Simons 1955 Model, *Phys. Rev. E*, **64**, 2001, 035104.

[20] Chakrabarti, S.: *Mining the Web: Discovering Knowledge from Hypertext Data*, Morgan Kaufmann Publishers, 2002.

[21] Chakrabarti, S., van der Berg, M., Dom, B.: Focused Crawling: a New Approach to Topic–Specific Web Resource Discovery, *Computer Networks*, **31**(11-16), 1999, 1623–1640.

[22] Cho, J., Garcia-Molina, H.: Effective Page Refresh Policies for Web Crawlers, *ACM Transactions on Database Systems*, **28**(4), 2003, 390–426.

[23] Clark, C. W., Mangel, M.: *Dynamic State Variable Models in Ecology: Methods and Applications*, Oxford University Press, New York, Oxford, 2000.

[24] Coleman Jr, H. J.: What Enables Self–Organizing Behavior in Businesses, *Emergence (A Journal of Complexity Issues in Organizations and Management)*, **1**(1), 1999, 33–48.

[25] Cong, J., Smith, M.: A Parallel Bottom–Up Clustering Algorithm with Applications to Circuit Partitioning in VLSI Design, in: *Annual ACM IEEE Design Automation Conference, Proceedings of the* 30^{th} *International Conference on Design Automation*, New York, NY, USA, 1993, 755–760.

[26] Csuhaj-Varjú, E.: Eco–Grammar Systems: Recent Results and Perspectives, in: *Artificial Life: Grammatical Models* (G. Păun, Ed.), Black Sea University Press, Bucharest, 1995, 79–103.

[27] Csuhaj-Varjú, E.: Networks of Language Processors, *EATCS Bulletin*, **63**, 1997, 120–134.

[28] Csuhaj-Varjú, E.: Networks of Language Processors: a Language Theoretic Approach to Filtering and Cooperation, in: *ERCIM Workshop Proceedings, Fifth DELOS Workshop on Filtering and Collaborative Filtering* (L. Kovács, Ed.), 1997, 91–97.

[29] Csuhaj-Varjú, E., Dassow, J.: On the Size of Components of Probabilistic Cooperating Distributed Grammar Systems, *Theory Is Forever*, 2004, 49–59.

[30] Csuhaj-Varjú, E., Dassow, J., Kelemen, J., Păun, G.: *Grammar systems. A Grammatical Approach on Distribution and Cooperation*, Gordon and Breach, London, UK, 1994.

[31] Csuhaj-Varjú, E., Kelemen, J., Kelemenová, A., Păun, G.: Eco–Grammar Systems: A Grammatical Framework for Studying Lifelike Interactions, *Artificial Life*, **3**, 1997, 1–28.

[32] Csuhaj-Varjú, E., Kelemen, J., Păun, G.: Grammar Systems with WAVE–like Communication, *Computers and Artificial Intelligence*, **15**(5), 1996, 419–436.

[33] Csuhaj-Varjú, E., Mitrana, V.: Dynamical Teams in Eco–Grammar Systems, *Fundamenta Informaticae*, **44**, 2000, 83–94.

[34] Csuhaj-Varjú, E., Salomaa, A.: Networks of Parallel Language Processors, in: *New Trends in Formal Languages, Control, Cooperation and Combinatorics. Lecture Notes in Computer Science* (G. Păun, A. Salomaa, Eds.), vol. 1218, 1997, 299–318.

[35] Dassow, J., Kelemen, J.: On Parallelism in Colonies, *Cybernetics and Systems: An International Journal*, **24**, 1993, 37–49.

[36] Dassow, J., Păun, G.: *Regulated Rewriting in Formal Language Theory*, Springer–Verlag, New York, NY, USA, 1996.

[37] Diligenti, M., Coetzee, F., Lawrence, S., Giles, C. L., Gori, M.: Focused Crawling Using Context Graphs, in: *Proceedings of the* 26^{th} *International Conference on Very Large Databases*, Morgan Kaufmann, San Francisco, CA, USA, 2000.

[38] Eppstein, D.: Fast Hierarchical Clustering and Other Applications of Dynamic Closest Pairs, *Journal of Experimental Algorithmics*, **5**(1), 2000, 1–23.

[39] Erdős, P., Rényi, A.: On Random Graphs, *Publ. Math., Debrecen*, **6**, 1959, 290–297.

[40] Erdős, P., Rényi, A.: The Evolution of Random Graphs, *Publ. Math. Inst. Hung. Acad. Sci.*, **5**, 1960, 17–61.

[41] Erdős, P., Rényi, A.: On the Strength of Connectedness of a Random Graph, *Acta Mathematica Scientia Hungary*, **12**, 1961, 261–267.

[42] Errico, L.: WAVE: An Overview of the Model and the Language, CSRG, Dept. Electronic and Electr. Eng., Univ. of Surrey, UK, 1993.

[43] Fahlman, S. E., Hinton, G. E., Seijnowsky, T. J.: Massively Parallel Architectures for AI: NETL, THISTLE and Boltzmann Machines, in: *Proc. AAAI National Conf. on AI*, William Kaufman, Los Altos, CA, USA, 1983, 109–113.

[44] Falconner, K.: *Fractal Geometry, Mathematical Foundations and Applications*, John Wiley and Sons, 1990.

[45] Fernau, H., Holzer, M., Freund, R.: Hybrid Modes in Cooperating Distributed Grammar Systems: Internal versus External Hybridization, *Theoretical Computer Science*, **259**, 2001, 405–426.

[46] Flenner, R., Abbott, M., Boubez, T., Cohen, F., Krishnan, N., Moffet, A., Ramamurti, R., Siddiqui, B., Sommers, F.: *Java P2P Unleashed: with JXTA, Web Services, XML, Jini^{TM}, JavaSpaces and J2EE*, Sams Publishing, Indianapolis, IN, USA, 2001.

[47] von Foerester, H.: On Self–Organising Systems and Their Environments, in: *Self–Organising Systems* (M. C. Yovits, S. Cameron, Eds.), Pergamon Press, London, UK, 1960, 30–50.

[48] Frank, E., Paynter, G. W., Witten, I. H., Gutwin, C., Nevill-Manning, C. G.: Domain–Specific Keyphrase Extraction, in: *Proceedings of the* 16^{th} *International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, Los Altos, CA, USA, 1999, 668–673.

[49] Fryxell, J. M., Lundberg, P.: *Individual Behavior and Community Dynamics*, Chapman and Hall, New York, NY, USA, 1998.

[50] Garfield, E.: It's a Small World After All, *Current Contents*, **43**, 1979, 5–10.

[51] Girvan, M., Newman, M. E. J.: Community Structure in Social and Biological Networks, *Proc. Natl. Acad. Sci. USA*, **99**(12), 2002, 7821–7826.

[52] Giuli, T. J., Maniatis, P., Baker, M., Rosenthal, D. S. H., Roussopoulos, M.: Attrition Defenses for a Peer–to–Peer Digital Preservation System, in: *Proceedings of the USENIX Annual Technical Conference, General Track*, 2005, 163–178.

[53] Guare, J.: *Six Degrees of Separation*, Vintage Books, New York, NY, USA, 1990.

[54] Haykin, S.: *Neural Networks: A Comprehensive Foundation*, Prentice Hall, New Jersey, USA, 1999.

[55] Heylighen, F.: Self–Organization, Emergence and the Architecture of Complexity, in: *Proceedings of the* 1^{st} *European Conference on System Science, (AFCET, Paris)*, 1989, 23–32.

[56] Heylighen, F.: The Science of Self–Organisation and Adaptivity, in: *The Encyclopedia of Life Support Systems*, UNESCO Publishing–Eolss Publishers, 2002.

[57] Hillis, W. D.: *The Connection Machine*, The MIT Press, Cambridge, MA, USA, 1985.

[58] Hopcroft, J. E., Ullman, J. D.: *Introduction to Automata Theory, Languages and Computation*, Addison–Wesley, Reading, MA, Menlo Park, CA, London, Amsterdam, Don Mills, Ontario, Sydney, 1979.

[59] Hung, C., Wermter, S., Smith, P.: Predictive Top–Down Knowledge Improves Neural Exploratory Bottom–Up Clustering, in: *ECIR 2004, LNCS 2997* (S. McDonald, J. Tait, Eds.), Springer–Verlag, Berlin–Heidelberg, 2004, 154–166.

[60] Hutchison, D., Katz, R. H., Eds.: *Self–Organizing Systems, Second International Workshop, IWSOS 2007, The Lake District, UK, September 11–13, 2007, Proceedings*, vol. 4725 of *Lecture Notes in Computer Science*, Springer, 2007.

[61] Jajodia, S., Samarati, P., Sapino, M. L., Subrahmanian, V. S.: Flexible Support for Multiple Access Control Policies, *ACM Transactions on Database Systems*, **26**(2), June 2001, 214–260.

[62] Joachims, T.: A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization, in: *Proceedings of ICML–97,* 14^{th} *International Conference on Machine Learning*, Morgan Kaufmann Publishers, San Francisco, CA, USA.

[63] Kampis, G.: *Self–Modifying Systems in Biology and Cognitive Science: A New Framework for Dynamics, Information and Complexity*, Pergamon, Oxford, UK, 1991.

[64] Kant, K., Iyer, R.: A Performance Model for Peer–to–Peer File–Sharing Services, *WWW–11 Poster Session*, May 2002.

[65] Kant, K., Iyer, R. K., Tewari, V.: A Framework for Classifying Peer–to–Peer Technologies, in: *Proceedings of the* 2^{nd} *IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID 2002)*, 2002, 368–375.

[66] Karinthy, F.: *Everything is Different*, chapter Chains, Budapest, 1929.

[67] Kauffman, S.: *The Origins of Order: Self–Organization and Selection in Evolution*, Oxford University Press, New York, NY, USA, 1993.

[68] Kauffman, S.: *At Home in the Universe: the Search for the Laws of Self– Organization and Complexity*, Oxford University Press, New York, NY, USA, 1995.

[69] Kochen, M., Ed.: *The Small World*, Ablex, Norwood, NJ, USA, 1989.

[70] Kohonen, T.: *Self–Organizing Maps*, vol. 30 of *Springer Series in Information Sciences*, Springer, Berlin, Heidelberg, New York, 2001.

[71] Kókai, I., Lőrincz, A.: Fast Adapting Value Estimation Based Hybrid Architecture for Searching the World–Wide Web, *Applied Soft Computing*, **2**, 2002, 11–23.

[72] Lázár, K.: A Language Theoretic Approach to the Crawlers' Problem, in: *Proceedings of the* 5^{th} *International Workshop on Emergent Synthesis (IWES'04)* (K. Ueda, L. Monostori, A. Márkus, Eds.), 2004, 19–26.

[73] Lázár, K., Csuhaj-Varjú, E., Lőrincz, A.: Peer–to–Peer Networks: a Language Theoretic Approach, *Computing and Informatics*, **27**, 2008, 403–422.

[74] Lázár, K. A., Csuhaj-Varjú, E., Lőrincz, A.: On Eco–Foraging Systems, *submitted*.

[75] Lázár, K. A., Csuhaj-Varjú, E., Lőrincz, A.: A Combined Theoretical and Experimental Approach to Goal–Oriented Herds of Internet Crawlers, in: *Proceedings of the European Conference on Complex Systems (ECCS 2009)*, The University of Warwick, UK, 2009, page 134.

[76] Lázár, K. A., Csuhaj-Varjú, E., Lőrincz, A., Vaszil, G.: Dynamically Formed Clusters of Agents in Eco–Grammar Systems, *International Journal of Foundations of Computer Science*, **20**(2), 2009, 293–311.

[77] Lázár, K. A., Farkas, C.: Security in T_{ci}NMP Systems, in: *Proceedings of the* 5^{th} *International Workshop on Security in Information Systems (WOSIS 2007)*, INSTICC PRESS, Portugal, 2007, 95–104.

[78] Lőrincz, A., Lázár, K. A., Palotai, Z.: Efficiency of Goal–Oriented Communicating Agents in Different Graph Topologies: a Study with Internet Crawlers, *Physica A*, **378**(1), 2007, 127–134.

[79] Lőrincz, A., Gábor, B., Mandusitz, S., Palotai, Z.: Bottom–Up Clustering and Top–Down Shattering of Scale–Free Environments for Information Fusion, in: *Proceedings of the Seventh International Conference on Information Fusion* (P. Svensson, J. Schubert, Eds.), vol. I, International Society of Information Fusion, Mountain View, CA, USA, Jun 2004, 471–478.

[80] Lőrincz, A., Kókai, I., Meretei, A.: Intelligent High–Performance Crawlers Used to Reveal Topic–Specific Structure of the WWW, *Int. J. Founds. Comp. Sci.*, **13**, 2002, 477–495.

[81] MacKay, D. J. C.: *Information Theory, Inference and Learning Algorithms*, Cambridge University Press, Cambridge, UK, 2003.

[82] Malkhi, D.: Secret Sharing. An Advanced Course in Computer and Network Security, The Hebrew University of Jerusalem, `http://www.cs.huji.ac.il/~ns/SS.doc`, 2002.

[83] Mandelbrot, B. B.: *The Fractal Geometry of Nature*, W. H. Freedman and Co., San Francisco, CA, USA, 1982.

[84] Marchioria, M., Latora, V.: Harmony in the Small–World, *Physica A*, **285**, 2000, 39–56.

[85] Mataric, M.: Reinforcement Learning in the Multi–Robot Domain, *Autonomous Robots*, **4**(1), 1997, 73–83.

[86] McCallum, A., Nigam, K., Rennie, J., Seymore, K.: A Machine Learning Approach to Building Domain–Specific Search Engines, in: *Proceedings of the* 16^{th} *International Joint Conference on Artificial Intelligence*, 1999, 662–667.

[87] McCallum, A., Nigam, K., Rennie, J., Seymore, K.: Automating the Construction of Internet Portals with Machine Learning, *Information Retrieval*, **3**, 2000, 127–163.

[88] Menezes, A. J., Vanstone, S. A., Oorschot, P. C. V., Eds.: *Handbook of Applied Cryptography*, CRC Press, Boca Raton, FL, USA, 1996.

[89] Menzer, F., Pant, G., Srinivasan, P.: Topical Web Crawlers: Evaluating Adaptive Algorithms, *ACM Transactions on Internet Technology*, **4**(4), 2004, 378–419.

[90] Merton, R. K.: The Matthew Effect in Science, *Science*, **159**, 1968, 56–63.

[91] Milgram, S.: The Small World Problem, *Psychology Today*, **1**, 1967, 60–67.

[92] Mitrana, V.: Hybrid Cooperating/Distributed Grammar Systems, *Comput. Artif. Intell.*, **12**(1), 1993, 83–88.

[93] Nejdl, W., Siberski, W., Thaden, U., Balke, W.-T.: Top–k Query Evaluation for Schema–Based Peer–to–Peer Networks, in: *ISWC 2004, LNCS*, vol. 3298, Springer–Verlag, Berlin–Heidelberg, 2004, 137–151.

[94] Newman, M., Barabási, A. L., Watts, D. J.: *The Structure and Dynamics of Networks*, Princeton University Press, Princeton, NJ, USA, 2006.

[95] Newman, M. E. J.: The Structure and Function of Complex Networks, *SIAM Review*, **45**, 2003, 167256.

[96] Newman, M. E. J., Girvan, M.: Finding and Evaluating Community Structure in Networks, *Phys. Rev. E*, **69**, 2004, 026113.

[97] Nicolis, G., Prigogine, I.: *Self–Organization in Nonequilibrium Systems: From Dissipative Structures to Order through Fluctuations*, John Wiley, New York, NY, USA, 1977.

[98] Pachepsky, E., Taylor, T., Jones, S.: Mutualism Promotes Diversity and Stability in a Simple Artificial Ecosystem, *Artificial Life*, **8**(1), 2002, 5–24.

[99] Palotai, Z., Farkas, C., Lőrincz, A.: Is Selection Optimal in Scale–Free Small Worlds?, *ComPlexUs*, **3**, 2006, 158–168.

[100] Palotai, Z., Kandár, T., Mohr, Z., Visegrády, T., Ziegler, G., Arató, P., Lőrincz, A.: Value Prediction in HLS Allocation Problems Using Intellectual Properties, *Applied Artificial Intelligence*, **16**, 2002, 151–192.

[101] Palotai, Z., Mandusitz, S., Lőrincz, A.: Distributed Novel News Mining from the Internet with an Evolutionary News Forager Community, in: *Int. Joint Conf. on Neural Networks 2004, 26 – 29 July 2004, Budapest*, IEEE Operations Center, Piscataway, NJ, USA, 2004.

[102] Palotai, Z., Mandusitz, S., Lőrincz, A.: Computer Study of the Evolution of 'News Foragers' on the Internet, in: *Swarm Intelligence in Data Mining* (C. Grosan, A. Abraham, V. Ramos, Eds.), vol. 34, Springer–Verlag, Berlin, Heidelberg, 2006, 203–219.

[103] Pant, G., Srinivasan, P.: Link Contexts in Classifier–Guided Topical Crawlers, *IEEE Transactions on Knowledge and Data Engineering*, **18**(1), January 2006, 107–122.

[104] Pant, G., Srinivasan, P., Menczer, F.: Exploration versus Exploitation in Topic Driven Crawlers, in: *Proceedings of the Second International Workshop on Web Dynamics*, 2002.

[105] Pfleeger, C. P., Pfleeger, S. L.: *Security in Computing*, Prentice–Hall, Upper Saddle River, NJ, USA, 2006.

[106] Pinkerton, B.: Finding what People Want: Experiences with the WebCrawler, in: *First World Wide Web Conference*, Chicago, IL, USA, 1994.

[107] Păun, G.: On the Generative Capacity of Hybrid CD Grammar Systems, *J. Inform. Process. Cybernet.*, **EIK 30**(4), 1994, 231–244.

[108] Rennie, J., McCallum, A. K.: Using Reinforcement Learning to Spider the Web Efficiently, in: *Proceedings of ICML–99,* 16^{th} *International Conference on Machine Learning* (I. Bratko, S. Dzeroski, Eds.), Morgan Kaufmann Publishers, San Francisco, CA, USA, 1999, 335–343.

[109] Rozenberg, G., Salomaa, A.: *Mathematical Theory of L Systems*, Academic Press, New York, 1980.

[110] Rozenberg, G., Salomaa, A., Eds.: *Handbook of Formal Languages*, Springer–Verlag, Berlin–Heidelberg, 1997.

[111] Salomaa, A.: Probabilistic and Weighted Grammars, *Information and Control*, **15**, 1969, 529–544.

[112] Salomaa, A.: *Formal Languages*, Academic Press, New York, 1973.

[113] Sandhu, R. S., Samarati, P.: Access Control: Principles and Practice, *IEEE Communications*, **32**(9), 1994, 40–48.

[114] Sapaty, P. S.: The WAVE Paradigm, Internal Report 17/92, Department of Informatics, University of Karlsruhe, 1992.

[115] SDL Forum Society: What is SDL?, `http://www.sdl-forum.org/SDL`, 2009.

[116] Shalizi, C. R.: *Causal Architecture, Complexity and Self–Organization in Time Series and Cellular Automata*, Ph.D. Thesis, University of Wisconsin at Madison, 2001.

[117] Simon, H. A.: On a Class of Skew Distribution Functions, *Biometrika*, **42**(3–4), 1955, 425–440.

[118] de Sola Price, D. J.: Networks of Scientific Papers, *Science*, **149**, 1965, 510–515.

[119] de Sola Price, D. J.: A General Theory of Bibliometric and Other Cumulative Advantage Processes, *J. Amer. Soc. Inform. Sci.*, **27**, 1976, 292–306.

[120] Steinmetz, R., Wehrle, K., Eds.: *P2P Systems and Applications*, vol. 3485 of *LNCS*, Springer–Verlag, Berlin–Heidelberg, 2005.

[121] Sun Microsystems: JXTA Java^{TM} Standard Edition v2.5: Programmers' Guide, `http://www.jxta.org`, 2007.

[122] Sutton, R., Barto, A. G.: *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA, USA, 1998.

[123] Tadic, B.: Temporal Fractal Structures: Origin of Power Laws in the World Wide Web, *Physica A*, **314**, 2002, 278–283.

[124] Tanenbaum, A. S., van Steen, M.: *Distributed Systems: Principles and Paradigms*, Prentice–Hall, Upper Saddle River, New Jersey, 2002.

[125] Tesauro, G. J.: Temporal Difference Learning and TD–Gammon, *Communication of the ACM*, **38**, 1995, 58–68.

[126] Thinking Machine Coorporation: *Connection Machine Model CM–2 Technical Summary*, Thinking Machine Coorporation, Cambridge, MA, USA, 1987.

[127] Travers, J., Milgram, S.: An Experimental Study of the Small World Problem, *Sociometry*, **32**, 1969, 425–443.

[128] Wasserman, S., Faust, K.: *Social Network Analysis: Methods and Applications*, Cambridge University Press, Cambridge, UK, 1994.

[129] Watts, D. J., Strogatz, S.: Collective Dynamics of Small–World Networks, *Nature*, **393**, 1998, 440–442.

[130] Wiener, N.: *The Mathematics of Self–Organising Systems. Recent Developments in Information and Decision Processes*, Macmillan, New York, NY, USA, 1962.

[131] Wijesekera, D., Jajodia, S., Parisi-Presicce, F., Hangstom, A.: Removing Permissions in the Flexible Authorization Framework, *ACM Transactions on Database Systems*, **28**(3), 2003, 209–229.

[132] Yule, G. U.: A Mathematical Theory of Evolution, based on the Conclusions of Dr. J. C. Willis, F.R.S., *Philosophical Transactions of the Royal Society of London, Ser. B*, **213**, 1925, 21–87.

[133] Ziegler, G., Farkas, C., Lőrincz, A.: A Framework for Anonymous but Accountable Self–Organizing Communities, *Information and Software Technology*, **48**(8), 2006, 726–744.